

NASIS DATA BASE MANAGEMENT SYSTEM - IBM 360/370 OS MVT IMPLEMENTATION

III - DATA SET SPECIFICATIONS

NASA-CR-134472) NASIS DATA BASE
MANAGEMENT SYSTEM - IBM 360/370 OS MVT
IMPLEMENTATION. 3: DATA SET
(Neoterics, Inc., Cleveland, Ohio.)
202 p HC

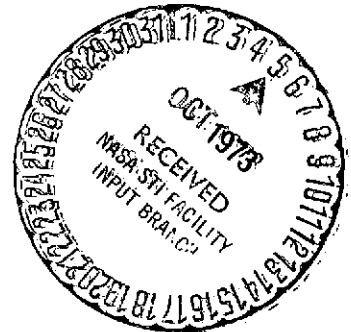
N73-31134

Unclas
13772

CSCL 09B G3/08

NEOTERICS, INC.

prepared for



NATIONAL AERONAUTICS AND SPACE ADMINISTRATION

Reproduced by
NATIONAL TECHNICAL
INFORMATION SERVICE
US Department of Commerce
Springfield, VA. 22151

NASA Lewis Research Center

Contract NAS 3-14979

1

1. Report No. NASA CR-134472		2. Government Accession No.		3. Recipient's Catalog No.	
4. Title and Subtitle NASIS DATA BASE MANAGEMENT SYSTEM - IBM 360/370 OS MVT IMPLEMENTATION III - DATA SET SPECIFICATIONS				5. Report Date September 1973	
				6. Performing Organization Code	
7. Author(s)				8. Performing Organization Report No. None	
9. Performing Organization Name and Address Neoterics, Inc. 2800 Euclid Avenue Cleveland, Ohio 44115				10. Work Unit No.	
				11. Contract or Grant No. NAS 3-14979	
12. Sponsoring Agency Name and Address National Aeronautics and Space Administration Washington, D.C. 20546				13. Type of Report and Period Covered Contractor Report	
				14. Sponsoring Agency Code	
15. Supplementary Notes Final Report. Project Manager, Charles M. Goldstein, Computer Services Division, NASA Lewis Research Center, Cleveland, Ohio					
16. Abstract The NASIS development workbook contains all the required system documentation. The workbook includes the following seven volumes: I - Installation Standards (CR-134470) II - Overviews (CR-134471) III - Data Set Specifications (CR-134472) IV - Program Design Specifications (CR-134473) V - Retrieval Command System Reference Manual (CR-134474) VI - NASIS Message File (CR-134475) VII - Data Base Administrator User's Guide (CR-134476)					
17. Key Words (Suggested by Author(s))				18. Distribution Statement Unclassified - unlimited	
19. Security Classif. (of this report) Unclassified		20. Security Classif. (of this page) Unclassified		21. No. of Pages 201	

TABLE OF CONTENTS

TOPIC A - MT/T

A.1	MODTAB - Transient Module Name File.	4
A.2	ESDTAB - External Symbol File.	6

TOPIC B - DATA BASE EXECUTIVE

B.1	DBPL/I Diagnostics	8
B.2	DBPL/I-DBPAC Interface	14
B.3	DBPAC Error Codes.	17
B.4	Mainline File Control Block.	24
B.5	List (SCB) Structure	28
B.6	List Error Control Block	29
B.7	Descriptor Descriptor File	31
B.8	DBPL/I-DBLIST Interface.	59
B.9	Sets Information File.	61
B.10	Set File	62

TOPIC C - UTILITIES

C.1	NASIS.USERIDS.	63
-----	------------------------	----

TOPIC D - MAINTENANCE

D.1	DBLOAD Error Codes Table	64
D.2	TRNSCT Descriptors	65
D.3	CORRECT Data Display Format.	67
D.4	DBLOAD Error Data Set.	69
D.5	Inverted Index Format.	70
D.6	Descriptor Editor Data Display Format.	72
D.7	Descriptor Editor Field Name Display Format.	74
D.8	DBLOAD Input Data Set.	76
D.9	Descriptor Editor Listing Format	77
D.10	INVERT Restart File.	79
D.11	INVERT SORTIN File	80
D.12	INVERT SORTOUT File.	81
D.13	INVERT PLEX File	82
D.14	INVERT RANGE File.	83
D.15	Descriptor Editor Checkpoint	84
D.16	MERGE INDEX File	86
D.17	Descriptor Editor REVIEW Display Format.	87
D.18	Descriptor Editor DEFIELD Structure.	90
D.19	Descriptor Editor DERECSEC Structure	92
D.20	Descriptor Editor DESECUR Structure.	94
D.21	Descriptor Editor DESUPER Structure.	96
D.22	Descriptor Editor DEVALID Structure.	98
D.23	Descriptor Editor DEFILD Structure.	100
D.24	Descriptor Editor DEXINIT.	103
D.25	Descriptor Editor LEX Structure.	104
D.26	Descriptor Editor LEHDR Structure.	113

D.27	CARDIN Freeform Parameter File	115
D.28	PRTOUT Print File.	116

TOPIC E - TERMINAL SUPPORT

E.1	TSPL/I Diagnostics	117
E.2	Terminal Control Block	120
E.3	TSTEXT-TCB Declaration	124

TOPIC F - DATA RETRIEVAL

F.1	RETDATA - Retrieval Data Tble.	128
F.2	EXPAND Display Format.	129
F.3	SELECT Display Format.	131
F.4	DISPLAY Display Format	133
F.5	PARSED Table.	137
F.6	SETS Display Format.	141
F.7	Retrieval Transient-Module Interface	143
F.8	PRINT Data Set Format.	146
F.9	EXPTAB - Expand Term Table	150
F.10	FLDTAB - Field Name Table.	152
F.11	FORMATS Display Format	156
F.12	SETAB - Sets Table	157
F.13	USERTAB - User Data Table.	159
F.14	EXPLAIN Display Format	162
F.15	SPRNTAB - Print Parameters	163
F.16	SEQ_FORM - Sequential Format Table	165
F.17	VALUTAB - Linear Search Value Table.	167
F.18	SRCHTAB - Linear Search Table.	169
F.19	COL_FORM - Columnar Format Table	172
F.20	FIELDS Display Format.	174
F.21	S\$ENTRY - Pseudo-set Expression Table.	175
F.22	ENTRYDEF - Pseudo-set Information Table.	176

TOPIC G - USAGE STATISTICS

G.1	STATIC Descriptors	179
G.2	Maintenance Report Format.	188
G.3	Retrieval Report Format.	190
G.4	Snapshot Report Format	192

TOPIC H - IMMEDIATE COMMANDS

H.1	NASIS Message File	194
H.2	Strategy Data Set.	195
H.3	Strategy Display Format.	196
H.4	Strategy Names Display Format.	197
H.5	User Profile Table	198
H.6	User Profile Data Set.	199
H.7	VERBTAB - Command Table.	200

TOEIC A.1 - MT/T

A. DATA SET NAME:

MODTAB - Module Table File

B. CREATED BY:

DBMTAB - Utility Module

C. TYPE OF FILE:

Non-Data Base File

D. ORGANIZATION:

Sequential

E. KEY IDENTIFIER:

Not Applicable

F. RECORD LENGTH:

Fixed, 36 bytes

G. BLOCKING FACTOR:

360 bytes per block

H. PURPOSE:

This file is an ordered table of transient module information. Each record contains the entry point name used by the NASIS programs for the call-by-name of transient modules, the maximum time (in milliseconds) that a module can occupy a region of the overlay, the relative offsets of the module for each region (1, 2, or 3 in number), some control bytes allowed for to be used at execution time, and the segment numbers for the module in each of the regions. MODTAB file is read at NASIS initialization. The first record of this file contains control information concerning the rest of the file: the number of remaining MODTAB records and the number of regions processed; these counts are the first and second full-words of the first record. The other records are defined in the next section.

I. PL/I DECLARATION:

```
DECLARE
  1 MODATA,
```

```
/*A MODTAB RECORD */
```

```

2 MODULE CHAR(8),
2 TIME FIXED BIN(31,0),
2 OFFSET(3) CHAR(4),
2 X FIXED BIN(31,0),
2 Y CHAR(2),
2 W CHAR(1),
2 U CHAR(1),
2 SEGS CHAR(3),
2 END CHAR(1);

/*TRANSIENT MODULE NAME*/
/*MAXIMUM RESIDENT TIME*/
/*OFFSETS FOR 3 REGIONS*/
/*SOME CONTROL BYTES */
/* FOR MONITOR USAGE */
/* AT EXECUTION TIME */
/* ALLOWED FOR HERE */
/*BYTE / SEGMENT NUMBER*/
/*PAD TO FULL-WORD */

```

TOPIC A.2 - MT/T

A. DATA SET NAME:

ESDTAB - Composite ESD Table File

B. CREATED BY:

DBTABLE - Utility Module

C. TYPE OF FILE:

Non-Data Base File

D. ORGANIZATION:

Sequential

E. KEY IDENTIFIER:

Not Applicable

F. RECORD LENGTH:

Fixed, 32 bytes

G. BLOCKING FACTOR:

Unblocked

H. PURPOSE:

This file is a sorted table of all external symbols within the NASIS load module's composite ESD; it is generated by a stand-alone utility module DBTABLE. It is then processed by LEMTAB to generate the MODTAB file; ESDTAB is also used at execution time by the IBCALL module to perform the call-by-name function within NASIS. The first record in this file contains control information concerning the rest of the file: 2 full-words of zeroes (unused), 4 bytes containing the total number of remaining ESITAB records, 4 bytes containing the number of segment 1 type ESDTAB records within the file, and 4 full-words of zeroes (unused). The other records are defined in the next section.

I. PL/I DECLARATION:

DECLARE

1 ESICATA,	/*AN ESDTAB RECORD	*/
2 NAME CHAR(8),	/*EXTERNAL SYMBOL NAME	*/
2 TYPE CHAR(1),	/*TYPE OF SYMBOL	*/

```
2 OFFSET CHAR(3),      /*HEX. OFFSET IN MODULE*/
2 SEG# CHAR(1),         /*SEGMENT NUMBER      */
2 DATA CHAR(3),        /*OTHER ESD DATA      */
2 FILL CHAR(16);        /*    FAD (UNUSED)     */
```

TOHIC B.1 - DATA BASE EXECUTIVE

A. DATA SET NAME:

DBPL/I Diagnostics

B. CREATED BY:

DB Preprocessor Function

C. TYPE OF FILE:

(4) Table

D. ORGANIZATION:

Keyed List

E. KEY IDENTIFIER (CONTROL FIELD):

Each diagnostic comment has a five-character identification key having the form: DBnnn, where nnn is a unique identification number.

F. RECORD LENGTH:

Variable

G. BLOCKING FACTOR:

Not Applicable

H. PURPOSE:

DBPL/I Diagnostic Comments are generated into mainline source programs by the DB preprocessor function (see Section IV, Topic B.1 of the DWE).

I. DBPL/I DIAGNOSTIC COMMENTS:

DB001 INITIALIZATION COMPLETE.

Informative - the % INCLUDE DB statement has been successfully processed.

DB002 'DB' FOUND WITHOUT ARGUMENT. IT IS A RESERVED IDENTIFIER.

Severe error - a DB preprocessor function reference has no parenthesized argument.

DB003 MISSING LEFT PARENTHESIS.

Severe error - a DB preprocessor function reference does not begin with double left parentheses. Processing of this DB reference was abandoned because the closing right parenthesis would not be able to be found.

DB004 ARGUMENT ABANDONED. TOO MANY DBPL/I ERRORS.

Error - more than four errors have been noted from one DB preprocessor function reference so it is being abandoned. This diagnostic may arise when the right parenthesis are missing at the end of the argument (PL/I passes the remainder of the source program to the DB function).

DB005 EXTRANEIOUS TEXT IGNORED.

Error - if this message immediately follows DB009, then the statement has been processed properly but additional clause(s) other than comments intervene before the semicolon. Verify that the statement has its own semicolon. If this message follows a diagnostic other than DB009, it means merely that part of the statement was ignored.

DB006 MISSING SEMICOLON OR CCLON. 'text'

Severe error - the right parenthesis at the end of a DB preprocessor function reference has been encountered unexpectedly. The label or statement "text" shown was ignored.

DB007 MISSING SEMICOLON. 'text'

Severe error - the right parenthesis at the end of a DB preprocessor function reference has been encountered unexpectedly. The statement "text" shown was ignored.

DB009 DBPL/I STATEMENT: 'text; comments'

Informative - a non-null statement has been found. The "text" shown is the statement as internally rearranged into a standard format without embedded comments for further analysis. The "comments" shown

are those extracted from the statement.

DB011 STATEMENT FOLLOWS FINISH.

Severe error - the statement has been ignored because it follows the DB((FINISH;)) reference.

DB013 STATEMENT HAS 'n' MORE LEFT PARENTHESES THAN RIGHT PARENTHESES.

Severe error - the statement semicolon has been found but the parentheses are unbalanced. The statement was ignored.

DB015 UNKNOWN STATEMENT KEYWORD: 'word'.

Severe error - the 'word' shown is not the first word of a DBPL/I statement. The statement was ignored.

DB017 INVALID LISTERROR ACTION.

Severe error - an ON LISTERROR statement was ignored because its action clause was neither SYSTEM nor GO TO.

DB019 INVALID ERRORFILE.

Severe error - an ON ERRORFILE statement was ignored because its filename was longer than eight characters or was not terminated by a right parenthesis.

DB021 INVALID ON CONDITION.

Severe error - an ON statement was ignored because it was neither ON LISTERROR nor ON ERRORFILE. These are the only ON statements recognized by the DB preprocessor function.

DB023 MISSING LIST POINTER.

A GET LIST or PUT LIST statement was ignored because it did not contain a required parenthesized list pointer.

DB025 INVALID GET LIST CLAUSE.

Severe error - a GET LIST statement was ignored because it did not contain either a KEY(0) or a KEY INTO clause.

DB026 INVALID PUT LIST CLAUSE.

Severe error - a PUT LIST statement was ignored

because it did not contain required INTERNAL KEY FROM clauses.

DB027 INVALID FILE.

Severe error - a statement having a FILE clause was ignored because its filename was longer than eight characters or was not terminated by a right parenthesis.

DB028 MISSING LIST.

Severe error - a SET statement was ignored because it did not contain a required LIST clause.

DB029 MISSING FILE.

Severe error - a statement that should have a FILE clause was ignored because it did not have one.

DB030 MISSING SIZE.

Severe error - a SET LIST statement was ignored because it did not contain a required SIZE clause.

DB031 INVALID ON ACTION.

Severe error - an ON ERRORFILE statement was ignored because its action clause was neither SYSTEM nor GO TO.

DB032 MISSING 'LIST' OR 'KEY' CLAUSE.

Severe error - a GET FILE statement was ignored because it did not contain either a LIST or a KEY clause.

DB033 MISSING FIELD CLAUSE.

Severe error - a statement that should have a FIELD clause was ignored because it did not have one.

DB034 MISSING 'LIKE LIST'.

Severe error - a SET LIST SIZE statement was ignored because it did not contain a required LIKE LIST clause.

DB035 MISSING INTO.

Severe error - a GET FIELD statement was ignored because it did not contain a required INTO clause.

DB037 MISSING FROM.

Severe error - a PUT or REPUT statement was ignored because it did not contain a required FROM clause.

DB039 MORE ITEMS THAN FIELDS.

Severe error - excess INTO or FROM items in a GET, PUT or REPUT FIELD statement were ignored.

DB041 MORE FIELDS THAN ITEMS.

Severe error - excess fieldnames in a GET, PUT or REPUT FIELD statement were ignored because the INTO or FROM clause has too few items.

DB043 INVALID OPEN CLAUSE.

Severe error - an OPEN statement has been abandoned because one of its substatements has an invalid or out of order FILE, TITLE, access or function clause.

DB045 INVALID READ OPTION(S).

Severe error - a READ statement has been ignored because it has an invalid or out of order file-positioning or NOLOCK option.

DB047 INVALID CLOSE SYNTAX.

Severe error - a CLOSE statement has been abandoned because one of its substatements has an invalid or out of order FILE or ERASE clause.

DB049 MISSING FROM.

Severe error - a WRITE statement has been ignored because it did not contain a required FROM clause.

DB051 MISSING KEYFROM.

Severe error - a LOCATE statement has been ignored because it did not contain a required KEYFROM clause.

DB053 LIST OPTION MISSING.

Severe error - a FREE statement has been ignored because it did not contain a required LIST option.

DB055 INVALID LIST.

Severe error - a FREE LIST statement has been ignored because it did not contain a parenthesized set of list-pointers.

DB057 'filename' FILE HAS STATEMENT DIAGNOSTIC(S).

Severe error - the FINISH statement has not generated a Mainline File Control Block declaration (see Section III, Topic B.4 of the DWB) for the 'filename' shown because errors in its use have been previously detected. Note that a missing MFCB declaration will yield "undefined qualified name" diagnostics from the PL/I compiler for correct DBPL/I statements using the 'filename'.

DB059 'filename' FILE HAS NON-INPUT USE(S).

Informative - the 'filename' shown has a use that may conflict with the INPUT file function attribute.

DB061 'filename' FILE HAS NON-OUTPUT USE(S).

Informative - the 'filename' shown has a use that may conflict with the OUTPUT file function attribute.

DB063 'filename' FILE HAS NON-UPDATE USE(S).

Informative - the 'filename' shown has a use that may conflict with the UPDATE file function attribute.

DB065 'filename' FILE REQUIRES UPDATE ATTRIBUTE.

Informative - the 'filename' shown has a use that requires the UPDATE file function attribute, but this compilation does not contain a valid OPEN...UPDATE statement for the 'filename'.

DB067 'n' DBPL/I ERRORS.

Informative - the FINISH statement has been processed and 'N' errors were previously detected. The programmer should find and analyze the 'N' DBPL/I diagnostic comments.

TCHC B.2 - DATA BASE EXECUTIVE

A. DATA SET NAME:

DBPL/I - DBPAC Interface

B. CREATED BY:

DB Preprocessor Function

C. TYPE OF FILE:

(4) Table

D. ORGANIZATION:

Documentary Table

E. KEY IDENTIFIER (CONTROL FIELD):

Not Applicable

F. RECORD LENGTH:

Not Applicable

G. BLOCKING FACTOR:

Not Applicable

H. PURPOSE:

The DBPL/I - DBPAC Interface (see Table 1) specifies the MFCB.STATEMENT.OPERATION code, DBPAC entry point name, and argument types and order for the various DBPL/I statements and substatements. Thus, it serves to specify for the DB preprocessor function (see Section IV, Topic B.1 of the DWB) what MFCB assignments and CALL statements are to be generated for each DBPL/I statement. Conversely, it specifies for DBPAC (see Section IV, Topic B.2 of the DWB) what entry points will be entered and what and how information will be available at execution time for the performance of the various statement actions.

The various entry points and their argument types are declared by source code in SOURCE.IISRMAC member DBTEXT. Any program that includes the DB preprocessor also is given DBTEXT by an INCLUDE statement in DB.

TABLE 1.

ROUTINE	OPTION	OPERATION	ENTRYPOINT	ARG-1	ARG-2
SS CLOSE		00010000	DEPACFV	f	
SS CLOSE	ERASE	00011000	DBPACFV	f	
SE GET	FIELD	01100000	DEPACFV	f	v
SE GET	INTERNAL FLD	01100100	DEPACFV	f	r
GET	INDEX KEY	01100001	DEPACFV	f	v
GET	KEY SET	01100000	DEPACFP	p	v
GET	SUBFILE KEY				
	SET	01010001	DEPACFP	f	p
GET	LIST SET	01010000	DEPACFP	f	p
GET	INDEX LIST				
	SET	01010001	DEPACFP	f	p
GET	SUBFILE LIST				
	SET	01010010	DEPACFP	f	p
GET	RECORD	01000000	DEPACFP	f	r
LOCATE		11010000	DEPACFV	f	v
LOCATE	SUBFILE	11010010	DEPACFV	f	
SS OPEN		00100000	DEPACFV	f	
SE PUT	FIELD	10010000	DEPACFV	f	v
READ	KEY	11100000	DEPACFV	f	v
READ	KEY NOLOCK	11100100	DEPACFV	f	v
READ	INDEX KEY	11100101	DEPACFV	f	v
READ	SUBFILE KEY	11100010	DEPACFV	f	v
READ	SUBFILE KEY				
	NOLOCK	11100110	DEPACFV	f	v
READ	PER SUBFILE	11101010	DEPACFV	f	v
READ	PER SUBFILE				
	NOLOCK	11101110	DEPACFV	f	
READ	LIST	11101000	DEPACFP*	f	p
READ	LIST NOLOCK	11101100	DEPACFP*	f	p
READ	seq.	11110000	DEPACFV	f	
READ	seq. NOLOCK	11110100	DEPACFV	f	
READ	INDEX seq.	11110101	DEPACFV	f	
READ	SUBFILE seq.	11110010	DEPACFV	f	
READ	SUBFILE seq.				
	NOLOCK	11110110	DEPACFV	f	
READ	BACK	11111000	DEPACFV	f	
READ	BACK NOLOCK	11111100	DEPACFV	f	
READ	INDEX BACK	11111101	DEPACFV	f	
READ	SUBFILE BACK	11111101	DEPACFV	f	
READ	SUBFILE BACK				
	NOLOCK	11111110	DEPACFV	f	
SE REPUT	FIELD	10100000	DEPACFV	f	v
UNLOCK		11000000	DEPACFV	f	
UNLOCK	SUBFILE	11000010	DEPACFV	f	
WRITE		10000000	DEPACFP	f	r

SS = substatement
 SE = statement element
 f = filename
 p = list pointer
 r = record work area
 v = character string

*For READ LIST <NOLOCK> with the KEY (nn) clause, use entry point DEPACPF and a fullword subscript value as the third argument.

TOPIC B.3 - DATA BASE EXECUTIVE

A. DATA SET NAME:

DBPAC Error Codes

B. CREATED BY:

NDBPAC posts in HFCB.ERRCR.ONCCODE.

C. TYPE OF FILE:

(4) Table

D. ORGANIZATION:

Sequential

E. KEY IDENTIFIER (CONTROL FIELD):

Error-code

F. RECORD LENGTH:

Not Applicable

G. BLOCKING FACTOR:

Not Applicable

H. PURPOSE:

When any programs are written to interface with the NASIS system, they usually involve some type of interaction with the data base.

The data base executive was written with the intent of handling all such data base interfaces with the users of the NASIS system.

This is handled by the user writing a PL/I program and in it using the DBPL/I language extension to handle all of the input/output operations to the data base.

The data base executive has complete error detection and notification facilities built into it. Therefore, when the user's program is running, the data base executive will attempt to detect any and all errors which occur and communicate these errors back to the user's program.

The method of performing this communication of errors

is through the use of the data base executive error codes. These are fixed binary numbers which have unique meanings and are transmitted back to the user's program using the MFCB (mainline file control block) as a vehicle of communication. The various error codes which exist and their meanings are discussed in the following table:

ERROR

CODE EXPLANATION OF ERROR

01	Illegal attempt to imply open.
03	Tried to imply an open on a new file-name without the use of an open command.
20	Trying to open a file when the header descriptor's DESCOK switch is off.
21	Trying to open for COUTPUT or UPDATE and the file is not the anchor or an associate or a descriptor file.
22	End-of list (READ LIST Statement).
23	Number of files exceeds the number allowed in the MFCB file array.
25	The user is not the owner of the file, but he is attempting an open for UPDATE or OUTPUT.
26	Attempted open of the file for INPUT, the DATA switch indicates no data.
27	Open attempted but its function was not INPUT, OUTPUT, or UPDATE.
28	Open issued for UPDATE or COUTPUT was prohibited by the MNTNING, MNTNABLE, or the DATA switch.
29	Operation code unsupported.
30	Operation code error.
31	Key field failed general validation (READ KEY or LOCATE).
32	Key field failed specific validation (READ KEY or LOCATE).
34	Erase attempted on CLOSE but the file is not open for UPDATE.

- 35 Erase attempted on descriptor file other than the anchor.
- 36 The GET FIELD operation attempted but the last record operation was a LOCATE.
- 38 The GET FIELD operation attempted but there is no current record.
- 39 GET RECORD operation attempted but the user is not the owner.
- 40 GET LIST attempted, but file is not inverted index.
- 41 Key is null (READ KEY or LOCATE).
- 42 Key sequence error (LOCATE sequential).
- 43 Duplicate key error (LOCATE direct).
- 44 Not an COUTPUT file for WRITE.
- 46 No current record (PUT or REPUT).
- 47 Current record not locked (PUT or REPUT).
- 49 PUT or REPUT to INPUT file.
- 50 REPUT to non-UPDATE file.
- 51 REPUT following LOCATE.
- 52 GET operation (field is not in descriptor table).
- 53 Field failed general validation (PUT or REPUT).
- 54 Field failed special validation (PUT or REPUT).
- 55 Null value to be PUT.
- 56 Bit field too long (PUT or REPUT).
- 57 PUT to non-null bit switch.
- 58 PUT to non-null fixed length field.
- 59 PUT to non-null variable length (single element) field.
- 61 Field would make record too long (PUT).

- 62 Field would make record too long (REPUT).
- 63 Element would make record too long (PUT).
- 64 Element would make record too long (REPUT).
- 65 Element field too long (PUT or REPUT).
- 66 Too many (variable) elements (PUT).
- 67 No GET before REPUT (variable elements).
- 68 No good GET before REPUT (variable elements).
- 69 Undefined field (PUT or REPUT).
- 70 REPUT to never PUT (null) field (record not found).
- 71 Too many (fixed) elements (PUT).
- 72 (Fixed) element would make record too long (PUT).
- 73 No GET before REPUT (fixed elements).
- 74 No good GET before REPUT (fixed elements).
- 75 Field too long (PUT or REPUT).
- 76 Key would make cross reference record too long (PUT or REPUT).
- 77 Cross reference not found on record.
- 78 Target field 'actual' length checking indicates truncation.
- 79 Command system trying to open someone else's STATIC or TRNSCT data bases for UPDATE or OUTPUT.
- 80 Command system opening a data base (other than STATIC or TRNSCT) for either OUTPUT or UPDATE.
- 83 GET KEY incompatible with list.
- 84 GET KEY sequence error.
- 85 Field length less than 2 found. Data Base damage.
- 86 Field length beyond reclen found. Data Base damage.

- 87 Field length not equal to 2 plus a multiple of
element length found. Data Base damage.
- 88 Element length less than 1 found. Data Base
damage.
- 89 Element length beyond field length found. Data
Base damage.
- 90 Invalid DB2 header descriptor. DB1 descriptor or
damage.
- 91 Field descriptor reclen less than 78. Descriptor
damage.
- 92 Field length less than 2 in descriptor.
Descriptor damage.
- 93 Field length beyond reclen in descriptor.
Descriptor damage.
- 94 VALIDARG longer than 50 bytes would be truncated.
Descriptor damage.
- 95 SECURITY field length invalid. Should be 2 plus a
multiple of 8.
- 96 No descriptor found for key field. Descriptor
damage.
- 97 Invalid field length in index record found. Data
base damage.
- 98 Record missing from index region. Data base
damage.
- 99 End of data. (ISAM)
- 104 Keys equal - sequence error. (ISAM+100)
- 108 Key not found. (ISAM+100)
- 112 Keys out of sequence. (ISAM+100)
- 115 Keys do not coincide. (ISAM+100)
- 120 Keys coincide. (ISAM+100)
- 124 Invalid retrieval address. (ISAM+100)
- 128 Invalid record length. (ISAM+100)
- 131 Position past end of data set. (ISAM+100)

- 136 Position before start of data set. (ISAM+100)
- 140 Exceed maximum number of overflow pages. (ISAM+100)
- 144 Exceed maximum size of shared data set. (ISAM+100)
- 145 No data set-name found which is like the given one. (ISAM)
- 200 Attempt to (PUT or FEPUT) null pattern.
- 201 Attempt to (PUT or REPUT) to readonly field.
- 202 Undefined subfile or indexed field (READ, LOCATE, or UNLOCK).
- 203 Not an indexed field.
- 204 Not a subfile <control> field.
- 205 LOCATE to INPUT file.
- 206 No current anchor record (LOCATE SUBFILE).
- 207 Anchor record not locked (LOCATE SUBFILE).
- 208 No current subrecord (READ PER SUBFILE).
- 209 Anchor record not current (delete subrecord).
- 210 The file is not open (to pcst FLDTAB).
- 211 Descriptor damage detected while posting FLDTAB.
- 212 Anchor record not locked (delete subrecord).
- 213 Anchor record not parent of subrecord (delete subrecord).
- 214 Subrecord id not found in control field (delete subrecord).
- 215 Duplicate fixed length element (PUT or REPUT).
- 216 Duplicate varying length element (PUT or REPUT).
- 217 LOCATE SUBFILE not done because 131071 regions used.
- 218 NAMEFLD field length invalid. Should be 2 plus a multiple of 9.

- 219 GET superfield requires current subfile record.
- 220 RSECTYCE field length invalid. Should be 2 plus a multiple of 9.
- 221 Non-owner attempted to open associate having record level security.
- 222 Anchor dummy descriptor not found for associate or subfile field. Descriptor damage.

TCFIC 8.4 - DATA BASE EXECUTIVE

A. DATA SET NAME:

Mainline File Control Block

B. CREATED BY:

Declared by DB Preprocessor Function.

C. TYPE OF FILE:

(4) Table

D. ORGANIZATION:

linear structure followed by an array of linear structures.

E. KEY IDENTIFIER (CONTROL FIELD):

Within DBPAC the mainline file control block is known as a parameter named MFCB. Outside DBPAC each mainline file control block is an independent external controlled structure whose name is the DBPI/I file name (PLEX in the retrieval system). For this reason, file names must not conflict with other external name system. This file name is not padded with dollar signs the way a file title must be. The file name is passed as an argument in CALL statements to DBPAC and thus becomes the MFCB parameter.

F. RECORD LENGTH:

1324 bytes (hexadecimal 52C)

This is the length of the whole control block including the necessary dope vectors and a thirty-seven element array allowing up to thirty-seven data sets in a dataplex. The number of elements in the array may be adjusted, if necessary: - the control block size will be adjusted by 24 bytes per element and the MFCB.FILE.ARR_SIZE field must be suitably initialized but no changes are necessary in DBPAC or in other MFCB's.

G. BLOCKING FACTOR:

Not Applicable

H. PURPOSE:

The MFCB control block is used for communication between mainline programs and DBPAC. The declaration is generated by the DB preprocessor function with suitable initial value attributes. For DBPL/I statements in the mainline, the DB preprocessor function generates statements that post fields in the MFCB, such as the operation code. At execution time, the posted MFCB is passed as an argument to DBPAC. DBPAC performs the operation indicated in the MFCB, making reference to other fields in the MFCB as necessary and posting fields in the MFCB, such as MFCB.ERROR.ONCODE, which may subsequently be referenced in the mainline.

I. PL/I DECLARATION:

```

DECLARE
1 MFCB,                                /*MAINLINE FILE CONTROL BLOCK*/
  2 INITIALIZED BIT(2),                /*00: NEVER INITIALIZED      */
                                      /*10: INITIALIZED, CLOSED    */
                                      /*11: INITIALIZED, OPENED    */
  2 FILLER_1 BIT(6),                  /*NOT USED                   */
  2 STATEMENT,                        /*OR FUNCTION                 */
  3 OPERATION BIT(8),                 /*CODE                        */
  3 ONFIELD CHAR(8),                  /*FIELD NAME                  */
  2 FILLER_2 CHAR(3),                 /*NOT USED                    */
  2 ERROR,
  3 SYSTEM BIT(1),                   /*1: STANDARD DBPAC ACTION    */
                                      /*0: USER ERROR ROUTINE      */
                                      /*NOT USED                    */
  3 FILLER_3 BIT(7),
  3 ONCODE FIXED BINARY,
  3 ROUTINE LABEL,                   /*USER'S                      */
  3 ONRETURN LABEL,                  /*IN MAINLINE                 */
  2 FILE,
  3 ONFILE CHAR(8),                   /*FILE TITLE                  */
  3 OLFILE CHAR(8),                   /*TO SAVE FILE TITLE IF DYNAM*/
  3 OWNER_ID CHAR(8),                 /*OWNER OF THE FILE           */
  3 DSNAM CHAR(35),                   /*DATA SET NAME               */
  3 ATTRIBUTES
  4 ACCESS BIT(1),                   /*0: DIRECT                   */
                                      /*1: SEQUENTIAL               */
  4 SAVE_FUNC BIT(2),                 /*TO SAVE FUNCTION IF DYNAMIC*/
  4 FILLER_4 BIT(3),                 /*NOT USED                    */
  4 FUNCTION BIT(2),                  /*10: INPUT                   */
                                      /*01: OUTPUT                  */
                                      /*11: UPDATE                   */
  3 CURRENT_FILE FIXED BINARY, /*SUBSCRIPT IN FILE.ARRAY*/
  3 LAST_FILE FIXED BINARY,
                                      /*NUMBER OF FILES IN FILEPLEX OR DATAPLEX*/
  3 ARR_SIZE FIXED BINARY(15), /*DECLARED ARRAY SIZE        */
  3 ARRAY (37),

```

```

4 FILE_NAME CHAR(8),
4 DTP_POINTER,          /*DESCRIPTOR TABLE ADDRESS */
4 FCBP_POINTER,         /*FILE CONTROL BLOCK ADDRESS */
4 KYC_FIXED_BINARY(15), /*SUBSCRIPT OF KEY FIELD */
                        /*DESCRIPTOR IS ALWAYS =1. */
4 SWITCHES,
  5 CURRENT_BIT(1),
  5 LOCKED_BIT(1),
  5 WRITE_BIT(1),        /*FORCE WRITE */
  5 REWRITE_BIT(1),      /*FORCE REWRITE */
  5 ABSENT_BIT(1),       /*NULL OR SECURED RECORD */
  5 OPENED_BIT(1),       /*THE FILE IS OPEN */
  5 FILLER_5_BIT(2),     /*NOT USED */
4 FILLER_6_CHAR(1),     /*NOT USED */
4 RECORDCT_FIXED_BINARY(15); /* # OF USABLE DESC. */

```

J. DETAIL NOTES:

INITIALIZED - used entirely within DBPAC.

STATEMENT.OPERATION - see Section III, Topic E.2 of the DWB for the codes that are posted here by the DB preprocessor function.

STATEMENT.ONFIELD - posted by the DB preprocessor function.

ERROR.SYSTEM - posted by the DB preprocessor function.

ERROR.ONCODE - posted by DBPAC when an error is detected but not reset for successful operations. See Section III, Topic E.3 of the DWB for the DBPAC Error Codes.

ERROR.ROUTINE - posted by the DB preprocessor function.

ERROR.ONRETURN - posted by DBPAC when an error is detected.

FILE.ONFILE - posted by the DB preprocessor function. When the first character is not a pound sign indicating a descriptor file, DBPAC shifts the value one character to the right and posts a leading blank character.

FILE.OLFILE - used within DBPAC to detect need for reinitialization.

FILE.COWNER_ID - used within DBPAC.

FILE.DSNAME - used within DBPAC.

FILE.ATTRIBUTES.ACCESS and FUNCTION - posted by either the DB preprocessor function or, when in default, by DBPAC.

FILE.SAVE_FUNC - used within DBPAC.

FILE.CURRENT_FILE - used within DBPAC.

FILE.LAST_FILE - used within DBPAC.

FILE.ARR_SIZE - set by the DB preprocessor function indicating the dimension of the FILE.ARRAY.

FILE.ARRAY - this array is used within DBPAC. Each element of the array is a linear structure of fields relating to a data set. When the mainline is accessing a descriptor region or an inverted index, only the first element is used. Otherwise, the first element relates to the anchor data set and subsequent elements relate to associated and subfile and inverted index data sets.

FILE.ARRAY.FILE_NAME - the "title" of the data set having a leading blank or pound sign and a trailing blank or suffix.

FILE.ARRAY.DTP - the address of the (dynamically allocated) descriptor table for this data set.

FILE.ARRAY.FCBP - the address of the (dynamically allocated) file control block for this data set.

FILE.ARRAY.KYC - the subscript of the key field descriptor in the descriptor table array is always 1.

FILE.ARRAY.KYC(1) - one (the anchor) plus the number of associate data sets.

FILE.ARRAY.KYC(2) - KYC(1) plus the number of subfile data sets.

FILE.ARRAY.SWITCHES - switches used by DBPAC for the status of the data set.

FILE.ARRAY.RECORDCT - the number of descriptors in the descriptor table array. This number does not include descriptors of fields a given user does not have field security clearance to access.

TOHIC B.5 - DATA BASE EXECUTIVE

A. DATA SET NAME:

List SCB (Set Control Block) Structure

B. CREATED BY:

LBOSSET, Set Management

C. TYPE OF FILE:

(4) Table

D. ORGANIZATION:

Linear Structure

E. KEY IDENTIFIER (CONTROL FIELD):

Not Applicable

F. RECORD LENGTH:

Not Applicable

G. BLOCKING FACTOR:

Not Applicable

H. PURPOSE:

The SCB structure describes a particular set (or list) of data keys and points to the next SCB in the system chain (NULL if none exists).

I. PL/I DECLARATION:

DECLARE

1 SCB BASED(SCB_PNTR),	/*SET CONTROL BLOCK */
3 NXT_SCB POINTER,	/*POINTER TO NEXT SCB */
3 PARM_LIST,	/*ATTRIBUTES OF KEYS */
5 FLD_NAME CHAR(8),	/* FIELD NAME */
5 CON_RTN CHAR(8),	/* FORMATTING ROUTINE */
5 KEY_LNT BIN(15),	/* LENGTH IN LIST */
3 KEY_REC BIN(15),	/*NUMBER KEYS IN BUFFER*/
3 TOT_KEY BIN(15);	/*TOTAL NUMBER OF KEYS */

TOPIC B.6 - DATA BASE EXECUTIVE

A. DATA SET NAME:

LISTERR - List Error Control Block

B. CREATED BY:

Allocated by DBMTT.

Error fields are posted by DB preprocessor and DBOSET.
List chain anchors are initialized by DBMTT and posted by DBOSET.

C. TYPE OF FILE:

(4) Table

D. ORGANIZATION:

Simple structure

E. KEY IDENTIFIER (CONTROL FIELD):

Not applicable

F. RECORD LENGTH:

68 bytes (20 bytes data + 48 bytes PL/I dope vectors,
etc.)

G. BLOCKING FACTOR:

Not applicable

H. PURPOSE:

The list error control block holds the list segment chain anchors. (The chain used by the FREE all LISTS statement.) It also is the point of communication between DBLIST and mainline programs for list error handling when no MFCB is involved in the operation. (When an MFCB is involved in a list error situation, the MFCB is used for error indication, etc.)

I. PL/I DECLARATION:

DECLARE

```
1 LISTERR CTL EXT,          /*COMMON CONTROL BLOCK    */
  3 ERROR,                  /*
  5 SYSTEM BIT(1)           /*1: SYSTEM ACTION        */
```

```

INIT('1'B),
5 ONCODE FIXED BIN(15)
  INIT(0),

/*0: GC TO USER ERROR RTNE */
/*1: INVALID LIST OPERATION */
/*2: INCOMPATIBLE LISTS */
/* GET LIST KEY SET ERRORS: */
/*4: NULL INPUT LIST */
/*5: NO GET KEY SINCE RESET */
/*6: INCOMPATIBLE LISTS */
/* GET LIST KEY INTO ERROR: */
/*7: KEY SEQUENCE ERROR */
/*8: TRUNC, TARGET TOO SHORT*/
/* SET LIST LIKE LIST ERROR:*/
/*9: INVALID SIZE */
/* PUT LIST KEY FROM ERRORS:*/
/*10: NULL TARGET LIST */
/*11: WRONG LENGTH KEY VALUE*/
/*12: KEY SEQUENCE ERROR */
/*USER ERROR RTNE ADDRESS */
/*NULL: NO CHAIN */
/*ALLOCATOR MUST INITIALIZE */
/*FORWARD CHAIN ANCHOR */
/*BACKWARD CHAIN ANCHOR */

5 ROUTINE LABEL
3 PTR,

5(FIRST,
  LAST) PTR;

```

TOHC B.7 - DATA BASE EXECUTIVE

A. DATA SET NAME:

Data Base Descriptor File, ownerid. data-base.
data-base#

for example:

NASIS.ASRDI\$.ASRCIS#

where:

"ownerid" is the 1-8 character CS identification of the owner of the dataplex.

"data-base" is the 6 character data base name with the dollar sign character used for padding.

B. CREATED BY:

DBEDIT - the Descriptor Editor program

C. TYPE OF FILE:

(6) Data Base Descriptor file

D. ORGANIZATION

ISAM - Indexed Sequential Access Method, organized in one or more regions of contiguous records; one region for the anchor data set descriptors and, as necessary, a region for each associate, subfile and inverted index dataset's descriptors.

Records have the varying length universal record format (URF) built by DBPAC.

E. KEY IDENTIFIER (CONTROL FIELD):

The ISAM key length is 15 bytes consisting of:

7 character region name and 8 character FLDNAME.

A DBPL/I descriptor FILE is a contiguous set of records having the same region name. The DBPL/I FILE shall be OPENED using an 8 character TITLE value consisting of:

1 pound sign character (#) (signifying descriptor file).

6 character data base name with the dollar sign

character (\$) used for padding.

1 suffix character.

The suffix character shall be from the following ranges:

blank	anchor file descriptors
1-9	associate file descriptors
Z-Q	subfile descriptors
A-P	inverted index descriptors

DBPAC uses the data base name and suffix to automatically generate the region name value for the keys.

The DBPL/I KEY value is only the 8 character FIDNAME (name of the field being described) that completes the ISAM key value.

F. RECORD LENGTH:

34 bytes minimum for a file descriptor record.
78 bytes minimum for a field descriptor record.

G. BLOCKING FACTORS:

One 4096 byte page (block) will hold about 40 average descriptors; enough for a few regions for a simple data base. For a complicated data base with many data sets, fields, secondary fields, and security codes, three or more pages (blocks) may be required.

H. PURPOSE:

A data base descriptor file describes a data base in terms of the datasets, records, and fields the data base is composed of, and indicates their interrelationships. A data base descriptor file is created and maintained or modified by NDBEDIT, the descriptor editor, which is a system service program. Normally, a data base analyst runs NDBEDIT conversationally. NDBEDIT is responsible for producing a consistent and complete descriptor file according to this specification.

Then the data base may be loaded, maintained, retrieved, etc. by programs using NDBPAC for data base access. NDBPAC, when OPENing a data base, reads the descriptor file and from it builds a table that governs its further actions.

I. SAMPLE DATA BASE:

A sample data base is used to illustrate in detail how the descriptors shall describe a complicated data base. The sample data base consists of eight datasets related as shown in Figure 1.

The record layout in Figure 2 shows all eight record layouts in the sample data base for reference in the following sections of this specification.

J. DESCRIPTOR REGIONS:

The sample descriptor file consists of eight regions (having the suffixes ' ', 1, Z, Y, A, B, C, D) corresponding to the eight data sets in the sample data base. (Of course ISAM alphabetizes them ' ', A, B, C, D, Y, Z, 1.)

Each descriptor region has at least one file descriptor record and two field descriptor records (key and RECLEN).

Dummy descriptor records are required in anchor regions when the data base has associate and/or subfile datasets. They are also required in associate regions when subfile(s) are controlled from the associate dataset.

Figure 3 tabulates all the file, field, and dummy descriptors required to describe the sample data base.

K. FILE DESCRIPTOR RECORD:

The file descriptor record describes the dataset as a whole. It is uniquely identified by having a key (FLDNAME) of 8 blanks. It has the field values shown below. See Figure 4 for the field values of the sample descriptor file. All values should be posted except that if RECSECFP is NULL, RSECTYCD does not apply. The MXRECLEN field will appear if the DBRECL utility program has processed this file.

FIELD	VALUE	COMMENTS
-----	-----	-----
FLDNAME	8 blanks	DEPL/I key
DESCOK	OFF ON	incomplete descriptors. descriptors are complete.
FILETYPE	ANCHOR ASSOCIATE SUBFILE INDEX	type of data set being described

DESCRCT	numeric	number of field descriptors in this region. (The file descriptor is not to be counted.)
ESELNGTH	numeric	length in bytes of fixed portion of records including RECLen, key and fixed primary fields. For a spanned index, this includes the key suffix byte.
SPANNED	OFF ON	ordinary records. spanned records with internally suffixed keys.
DATA	OFF ON	no data on file yet. retrieval is possible.
MNTNABLE	ON OFF	maintenance is allowed. maintenance is prohibited.
MNTNING	OFF ON	no maintenance is in progress. maintenance is in progress.
LOADABLE	ON OFF	loading is allowed. loading this data set is prohibited.
RECSECFP	null numeric	records do not have a record security field. offset in bytes of record security field in records.
RSECTYCD	9 byte elements 8 alphanumeric 1 byte	zero or more record level security codes. record security password. mask for comparison with record security field.
MXRECLen	4 byte element	if exists, current maximum record length in file; only one element.

L. FIELD DESCRIPTOR RECORDS:

A field descriptor record indicates that a particular named field occurs in the dataset being described. It is uniquely identified within the region by having a key (FLDNAME) that is the name of the field. There are two kinds of field descriptors:

primary direct

secondary (direct and indirect)

All field descriptor records may have the values shown below:

FIELD -----	VALUE -----	COMMENTS -----
FLDNAME	8 alphanumerics blank padded	unique field name. DBPL/I key.
GENERCRT	8 alphanumerics blank padded	name of generic routine for testing input values.
VALIDRTN	8 alphanumerics blank padded	name of routine for testing and/or converting input values.
VALIDARG	0-50 bytes	argument to be supplied to VALIDRTN
NUMALIGN	OFF ON	string alignment, left jus- tification numeric alignment, right justification
REFORMAT	8 alphanumerics blank padded	name of routine for con- verting output values.
SECURITY	8 alphanumerics asterisk padded	0-18 field security pass- words

GENERCRT, VALIDRTN, VALIDARG, and NUMALIGN may even be posted for secondary (read only) fields because linear search, for example, may have to transform values to be used as comparands.

M. PRIMARY DIRECT FIELD DESCRIPTOR RECORDS:

A primary direct field descriptor record describes a maintainable field that occurs on each record of the dataset being described. There are five kinds of primary fields:

- single fixed bit
- single fixed byte
- single varying byte
- multiple fixed byte
- multiple varying byte

In addition to the field values shown in Section L, all primary descriptors have READONLY OFF and a selection of the following field values.

FIELD -----	VALUE -----	COMMENTS -----
READONLY	OFF	field value may be maintained (PUT and REPUT.)
VARFLD	FIXED	fixed length field in fixed portion of records. portion of records.
BITFLD	OFF ON	byte field bit field
FLDPOSIT	numeric	if VARFLD is FIXED, offset in bytes of field. if VARFLD is VARYING, relative field in variable portion of records.
FLDLEN	numeric	if BITFLD is ON, offset of bit (0,2,4 or 6) in byte specified by FLDPOSIT. if VARFLD is FIXED, internal length of field in bytes. if VARFLD is VARYING, maximum internal length of field in bytes with internal field length prefix.
ELTLIM	0 numeric	field does not have elements. maximum number of elements to be PUT into field or, for a control field, maximum number of sub- records per parent record.
ELTLEN	numeric	if VARELT is FIXED, internal length of elements in bytes. if VARELT is VARYING, maximum internal length of elements in bytes with internal element length prefix.
VARELT	FIXED VARYING	fixed length elements. varying length elements.
UNIQUELT	OFF ON	duplicate element values are allowed. internal element values must be unique.
INVFILE	alphabetic	descriptor region suffix for inverted index dataset. (null if none.)

INDEXEXT	OFF	index internal values of field.
	ON	index external values of field. (External length may require index key length greater than internal length.)

A single fixed bit field descriptor has:

VARFLD	FIXED
BITFLD	ON
FLDPOSIT	offset in bytes
FLDIEN	offset in bits (0,2,4 or 6)
INVFILE	null (may not be indexed)

See VEHAIKCD in the sample data base.

A single fixed byte field descriptor has:

VARFLD	FIXED
BITFLD	OFF
FLDPOSIT	offset in bytes
FLDLEN	internal length in bytes
INVFILE	optional if FLDLEN less than 254

See EMPPAYCL, EMPINSCL and VEHMAKE in the sample data base.

A single varying byte field descriptor has:

VARFLD	VARYING
FLDPOSIT	relative varying field
FLDLEN	maximum internal length including internal 2 byte field length prefix
ELTLIM	0 (zero)
INVFILE	optional if (FLDLEN-2) less than 254

See KIDNAME in the sample data base.

A multiple fixed byte field descriptor has:

VARFLD	VARYING
FLDPOSIT	relative varying field
FLDLEN	maximum internal field length including internal 2 byte field length prefix
ELTLIM	maximum number of elements
ELTLEN	internal element length in bytes
VARFLT	FIXED
UNIQUELT	optional
INVFILE	optional if ELTLEN less than 254

See the EMPKID and EMPVEH control fields in the sample data base.

A multiple varying byte field descriptor has:

VARFLD	VARYING
FLDPOSIT	relative varying field
FLDLEN	maximum internal field length including internal 2 byte field length prefix and internal 1 byte element length prefixes
ELTLIM	maximum number of elements
ELTLEN	maximum internal element length includ- ing internal 1 byte element length prefix. VARELT VARYING
UNIQUELT	optional
INVFILE	optional if (ELTLEN-1) less than 254.

See the KIDPET field in the sample data base.

CONTROL FIELD DESCRIPTORS

Every descriptor region must have a key descriptor for the field that uniquely identifies records in a dataset. It is a primary direct field descriptor record for a single fixed byte field.

SECURITY	must be null.
READONLY	is OFF
VARFLD	is FIXED
BITFLD	is OFF
FLDPOSIT	is 4
INVFILE	must be null

Each associate key descriptor is identical (except the region suffix) to the anchor key descriptor. See the EMPNAME field in the sample data base.

Each subfile dataset in a data base requires:

1. a control field in the anchor or an associate dataset
2. a separate descriptor region for the subfile dataset containing:
 - 2a. file descriptor record
 - 2b. RECLLEN secondary descriptor record
 - 2c. subrecord id key descriptor record
 - 2d. parent key secondary descriptor record
 - 2e. descriptor records for other subrecord fields.
1. A subfile control field descriptor describes a secondary multiple fixed byte field maintained by NDBPAC.

FLDNAME is a six-character name suffixed

by two blanks applying to the subfile.

GENERCRT	is IECVTID
VALIDRTN	is null
NUMALIGN	is CN
REFORMAT	is DBFMTID
SECURITY	is optional
READONLY	is CN
VAFELD	is VARYING
FLDPOSIT	is relative varying field
FLDLN	is maximum internal field length
ELTLN	is maximum number of elements.

Note that FLDLN or ELTLN limits the maximum number of subrecords per parent record.

ELTLN	is 3
VARELT	is FIXED
UNIQUELT	is CN
SUPCNTRL	is CN
SUBFILE	is alphabetic character descriptor region suffix for subfile dataset.
INVFILE	must be null

See EMPKID and EMPVEH in the sample data base.

2a. A subfile file descriptor record has:

FILETYPE	SUBFILE
----------	---------

2b. A subrecord RECLN descriptor is standard.

2c. A subrecord id key descriptor describes a primary single fixed byte field:

FLDNAME	is the six-character subfile name suffixed by "ID".
GENERCRT	is IECVTID
VALIDRTN	is null
NUMALIGN	is CN
REFORMAT	is DBFMTID
SECURITY	must be null
READONLY	is CFF
VAFELD	is FIXED
BITFLD	is CFF
FLDPOSIT	is 4
INVFILE	must be null

See EMPKIDID and EMPVEHID in the sample data base.

2d. A subrecord parent key descriptor describes a secondary single fixed byte field.

FLDNAME	is the six-character subfile name
---------	-----------------------------------

suffixed by "PK".
 GENERCRT, VALIDRTN, NUMALIGN, REFCRMAT and
 FLDLEN are the same as the anchor
 key descriptor.
 SECURITY is optional
 READONLY is CN
 VARFLD is FIXED
 BITFLD is CFF
 FLDPOSIT is 7
 INVFILE must be null

See EMPKIDPK and EMPVEHPK in the sample data base.

- 2e. Record level security may be independently specified for the anchor, associate and/or subfile datasets. It may not be specified for an inverted index dataset. Each dataset to have record level security must have:

RECSECFF field position of record security
 field
 RSECTYCE optional

in its file descriptor record and a primary direct field descriptor record for a single fixed byte field as shown in Figure 5 and as follows:

FLDNAME RECSEC suffixed by the descriptor
 region suffix (blnak for the
 anchor and a blank.
 GENERCRT DBCVTHX
 VAIDARG null
 NUMALIGN OFF
 REFCRMAT DBFMTHX
 SECURITY optional
 READONLY OFF
 VARFLD FIXED
 BITFLD OFF
 FLDPOSIT on anchor or associate datasets,
 after the key (ie. 4 + key FLDLEN +
 1). on subfile datasets, after the
 parent key field (ie. 4 + 3 +
 parent key FLDLEN + 1).
 INVFILE optional

See the RECSEC, RECSEC1, EMPKIDRS and EMPVEHRS fields in the sample data base.

N. SECONDARY READONLY FIELD DESCRIPTOR RECORDS:

A secondary field descriptor record describes a derived field made up of one or more component fields. There are two types: direct and indirect.

A direct secondary field descriptor redescibes part of all of one primary field or a field automatically maintained by NDBPAC such as RECLen and subfile control and parent key fields. A direct secondary descriptor has the same field values as a primary descriptor with the following qualifications:

READONLY is always ON. Field may only be retrieved (GET).

INVFILE must be null.

FLDPOSIT and FLDLEN will specify an internal location within or equal to a primary field.

Otherwise, the descriptor fields may specify a direct secondary field like any of the five types of primary fields. A secondary of the same type of length provides renaming and perhaps an alternate REFORMAT. A secondary "single fixed byte" with a shorter FLDLENGTH provides for subfields. A secondary "single varying byte" redefining a primary "multiple fixed byte" obtains the concatenation of the internal element values.

Every descriptor region shall have a secondary direct field descriptor for a single fixed byte RECLen as follows:

FLDNAME	is RECLen
GENERICRT	is DBCVTBI
VALIDRTN	is null
NUMALIGN	is ON
REFORMAT	is DBEMTRL
SECURITY	is optional
READONLY	is ON
VARFLD	is FIXED
BITFLD	is OFF
FLDPOSIT	is 0 (zero)
FLDLEN	is 4
INVFILE	must be null

(No dummy descriptors are used for RECLen. Direct secondary descriptors may be specified by the Data Base Analyst to provide unique field names for the various RECLens in a data base.)

An indirect secondary field descriptor describes a "superfield" make up of one or more primary or direct secondary component fields. No more than one of the component fields may be a multi-element field. The component field values will be concatenated in NAMEFLD order for retrieval. (If there is a multi-element

component field, then the superfield will yield multiple values.) An indirect secondary descriptor has the field values shown below.

FIELD -----	VALUE -----	COMMENTS -----
READONLY	CN	field may only be retrieved
NAMEFLD	9 byte elements hex '00'	one to 16 component field specifications use external value of component
	hex '80'	use internal value of component
	followed by 8 alphas	primary or direct secondary component fieldname blank padded
REFORMAT converting	8 alphas	name of routine for blank padded concatenated output value

If the component fields specified in NAMEFLD are all from the same dataset (anchor, associate or subfile), then the indirect secondary descriptor goes in the descriptor region for that dataset. (Dummy descriptor(s) are required for the indirect secondary descriptor if it is on an associate or subfile.) See the KIDID field in the sample data base.

If the components are from an associate file and from a subfile controlled from that associate file, then the indirect secondary descriptor goes in the descriptor region for that associate.

Otherwise the indirect secondary descriptor goes in the anchor descriptor region and no dummy descriptors are required. See the EMPTYPE field in the sample data base.

If any component is from a subfile dataset, the (1.) no components may be from any other subfile dataset and (2.) to GET such a field, a mainline program must first obtain a current record in the subfile -- NDBPAC will automatically ensure that the parent and associate record(s) are available when required.

Every anchor and associate descriptor region shall have a secondary indirect field descriptor for the key field, see Figure 6.

FLDNAME is 'FILEKEY '

READONLY is ON
 NAMEFLD has one element consisting of hex '00'
 followed by the name of the primary
 key field.
 REFORMAT is NULL.

(No dummy descriptor is used for FILEKEY on associate datasets.)

O. DUMMY DESCRIPTOR RECORDS

A dummy field descriptor indicates that the field occurs in an associate or subfile data set and if it has an inverted index data set. It has the field values shown below.

FIELD	VALUE	COMMENTS
-----	-----	-----
FLDNAME	8 alphanumerics blank padded	field name. DBPL/I key.
ASSOCFIL	numeric character	descriptor region suffix for associate data set. (Null if none.)
SUBFILE	alphabetic character	descriptor region suffix for subfile data set. (Null if none.)
SUBCNTRL	OFF ON	dummy descriptor for subfile field. primary (dummy if ASSOCFIL is posted) descriptor for sub- file control field.
INVFILE	alphabetic character	descriptor region suffix for inverted index data set. (Null if none.)

The descriptor file, shown graphically in Figure 7 and 8, is so designed that the anchor region describes the anchor records and also has dummy descriptors for all other fields on associate (1) or subfile (Z,Y) records thus indicating the presence of all associate and subfile data sets. It also indicates the presence of all inverted indexes for the data base (A,B,C,D).

An associate region (1) describes a data set of associate records and has dummy descriptors for all other fields on subfiles (Y) depending on the associate record. It also indicates the presence of all inverted indexes for the associate and dependent subfiles

(C,D).

A subfile region (Z) describes a data set of subfile records and indicates the presence of all inverted indexes for the subrecords (B).

An index region (A) describes a data set of inverted index records.

This all enables DBPAC to access a whole data base, an associate portion of a data base, a subfile, or an inverted index as if it were a degenerate case of a data base. This can be advantageous for certain utility functions.

P. INVERTED INDEX DESCRIPTORS:

If INVFILE is posted for a primary direct field, then a separate descriptor region must exist for the inverted index dataset. It contains only the following:

FILE DESCRIPTOR RECORD

FILETYPE is INDEX
 SPANNED is optional
 BSELENGTH is 4 + index key FLDLEN (+1 if SPANNED).

RECLEN SECONDARY DIRECT FIELD DESCRIPTOR RECORD

Single Fixed Byte

INDEX KEY SECONDARY DIRECT FIELD DESCRIPTOR RECORD

Single Fixed Byte

FLDNAME is same as indexed FLDNAME
 READONLY is ON
 FLDPOSIT is 4

If indexed field descriptor has INDEXEXT OFF, the FLDLEN is maximum internal indexed field value length (without 2 byte internal field length or 1 byte internal element length) and REFORMAT is same as indexed field REFORMAT. If indexed field descriptor has INDEXEXT ON, then FLDLEN is maximum external indexed field value length and REFORMAT is null or a blank stripper. In either case, FLDLEN does not include the internal "sequence number" suffix if SPANNED.

CROSS REFERENCES SECONDARY DIRECT FIELD DESCRIPTOR

Record Multiple Fixed Byte

FLDNAME is same as indexed field's record key
FLDNAME.

READONLY is CN.

FLDPOSIT is 1.

FLDLN is 4000.

ELTLN is 4000.

ELTLN is same as indexed field's record key
FLDLN.

REFORMAT is same as indexed field's record key
REFORMAT.

O. FILE AND FIELD DESCRIPTOR RECORD FORMATS:

File Descriptor Record Format

1	RECLN	0-3	Fixed	Binary	Length of header record, in bytes, including itself.
2	KEY	4-18	Fixed	EBCDIC	Identifier for this descriptor. Contains file name .
3	FILENAME	4-10	Fixed	EBCDIC	Seven-character file name for this descriptor. Contains data base and suffix.
4	DATAPLEX	4-9	Fixed	EBCDIC	Dataplex name padded with \$s to 6 characters.
5	SUFFIX	10	Fixed	EBCDIC	Identifier dataset.
6	FLDNAME	11-18	Fixed	EBCDIC	Contains blanks.
7	FILETYPE	19	Fixed	EBCDIC	1: Anchor 2: Associate 3: Subfile 4: Inverted index
8	DESCRCT	20-21	Fixed	Binary	Number of field descriptors for this dataset.
9	BSELNGTH	22-23	Fixed	Binary	Length of fixed portion of record.
10	DESCOK	24.0	Fixed	Bit	0: incomplete descriptors.

					1: complete descriptors.
11	SPANNED	24.2	Fixed	Bit	Applicable if FILETYPE=4: 0: ordinary records 1: spanned records with internally suffixed keys.
12	DATA	24.4	Fixed	Bit	0: No data on file. 1: Retrieval is possible.
13	MNTNABLE	24.6	Fixed	Bit	0: Maintenance prohibited. 1: Maintenance allowed.
14	MNTNING	25.0	Fixed	Bit	0: Maintenance not in progress. 1: Maintenance in progress.
15	-----	25.2	Fixed	Bit	Currently not applicable-Null. 1: Check record security.
16	LOADABLE	25.4	Fixed	Bit	0: Loading prohibited. 1: Loading allowed.
17	-----	25.6	Fixed	Bit	Currently not applicable-Null.
18	REMAINS	26-29	Fixed	-----	Currently not applicable-Null.
19	RECSECFP	30-31	Fixed	Binary	Record security field offset in records; null if none.
20	RSECTYCD	VrFld1	Var	-----	0-18 record security specifications consisting of a NASIS-id padded with \$s to 8 characters and a one byte mask.
21	MXRECLN	VrFld2	Var	-----	Maximum record length in file; field may not exist.

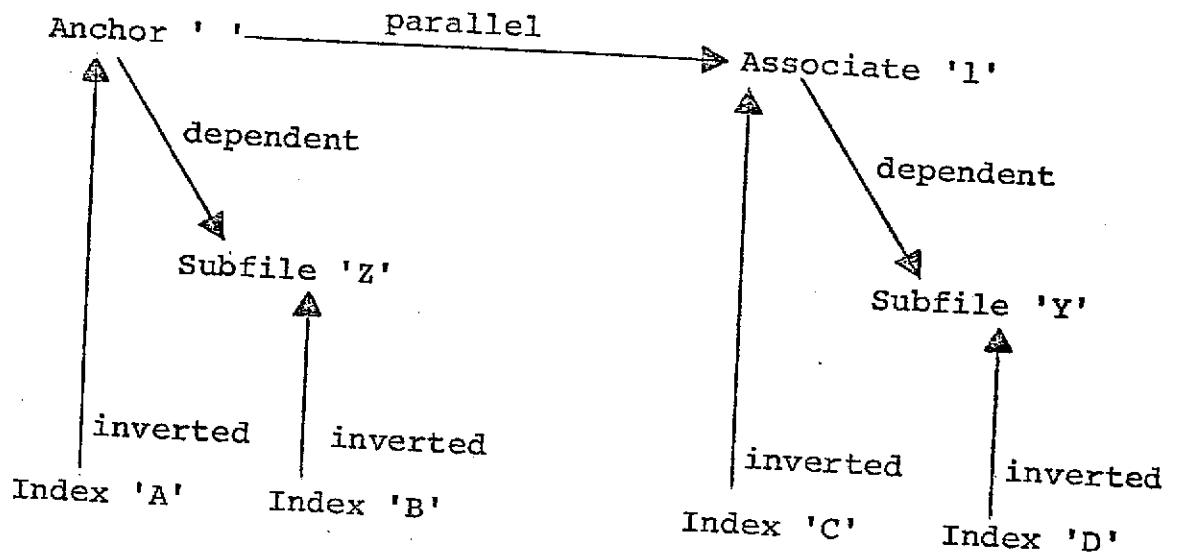
Field Descriptor Record Format

1	RECLEN	0-3	Fixed	Binary	Length of entire descriptor, in bytes, including itself.
2	KEY	4-18	Fixed	EBCDIC	Identifier for this descriptor. Contains file and field names.
3	PLENAME	4-10	Fixed	EBCDIC	Seven character file name for these descriptors.
4	FLDNAME	11-18	Fixed	EBCDIC	Name of field within file.
5	ASSOCFIL	19	Fixed	EBCDIC	Suffix of associated linear file which contains this field; null if none.
6	SUBFILE	20	Fixed	EBCDIC	Suffix of subfile which contains this field or which is controlled by this field; null if none.
7	INVFILE	21	Fixed	EBCDIC	Suffix of inverted file which indexes this field; null if none.
8	READONLY	22.0	Fixed	Bit	0: (Re)Put allowed. 1: (Re)Put prohibited.
9	SUBCNTRL	22.2	Fixed	Bit	Applicable if SUBFILE is non-null: 0: Field is on subfile. 1: This is control field.
10	VARFLD	22.4	Fixed	Bit	0: Fixed length field. 1: Varying length field.
11	BITFLD	22.6	Fixed	Bit	Applicable if VARFLD=0 0: Byte field. 1: Bit field.

12	NUMALIGN	23.0	Fixed	Bit	0: String (left) align. 1: Numeric (right) align.
13	VARELT	23.2	Fixed	Bit	Applicable if ELTLIM>0: 0: Fixed length elements. 1: Varying length elements.
14	UNIQUELT	23.4	Fixed	Bit	Applicable if ELTLIM>0: 0: Duplicate elements allowed. 1: Duplicate elements prohibited.
15	INDEXEXT	23.6	Fixed	Bit	Applicable if INVFILE is non-null. 0: Index internal values. 1: Index external values.
16	GENERCRT	24-31	Fixed	EBCDIC	Name of routine to be used for testing type of input characters (numeric, alpha, etc.); null if none.
17	VALIDBRN	32-39	Fixed	EBCDIC	Name of routine to be used for special validation or conversion of input data; null if none. Uses argument in VALIDARG.
18	REFORMAT	40-47	Fixed	EBCDIC	Name of routine to be used for any necessary output reformatting or conversion; null if none.
19	SPARE	48-55	Fixed	-----	Currently not applicable-Null.
20	NAMECNT	56-57	Fixed	-----	Currently not applicable-Null.
21	FLDPOSIT	58-59	Fixed	Binary	If VARFLD=0: byte

					offset in record. If VARFLD=1: relative varying field.
22	FLDLN	60-61	Fixed	Binary	If BITFLD=1: bit offset in byte. If VARFLD=0: field length in bytes. If VARFLD=1: maximum field length in bytes including 2 byte length indicator.
23	DFLDLEN	62-63	Fixed	-----	Currently not applicable-Null.
24	ELTLIM	64-65	Fixed	Binary	Applicable if VARFLD=1: 0: not multi-element. >0: maximum number of elements allowed.
25	DELTIM	66-67	Fixed	-----	Currently not applicable-Null.
26	ELTLEN	68-69	Fixed	Binary	If VARELT=0: element length in bytes. If VARELT=1: maximum element length in bytes including 1 byte length indicator.
27	DELTLEN	70-71	Fixed	-----	Currently not applicable-Null.
28	VALIDARG	VrFld1	VarHex		Argument to be used with VALIDRTN (test mask, limit, etc.). Fifty bytes maximum. Null if none.
29	NAMEFLD	VrFld2	Var		0-18 Superfield components consisting of a one byte function code (80x: external field element, 00x: internal field element) and an 8 character component field name.

30	SECURITY	VrFld3 Var	EBCDIC	0-18 field security codes consisting of a NASIS-id padded with *s to 8 characters.
----	----------	------------	--------	---



(Note: A simple data plex consists of only an anchor data set. A more complicated dataplex than the sample may have multiple index, associate, and/or subfile datasets, but the principles for describing it are shown in the sample.)

FIGURE 1. SAMPLE DATAPLEX

BYTLS
8 - double word
4 - word
2 - halfword
1 - 2 packed-decimal digits

IBM System/360 Record Layout Worksheet

INTERNATIONAL BUSINESS MACHINES CORPORATION
GX20-1711-0 U/M 025
Printed in U.S.A. (Rept. 4/70)

52

Record Name
SAMPLE DATAPLEX

02/07/72

FOLDOUT FRAME
2

APPLICATION

Page 2 of 2
Date

ANCHOR
FOLDOUT FRAME

RECLEN EMPNAME (KEY) BRECSEC EMPAYCL EMPKID (CONTROL FIELD)

HEXADECIMAL
DECIMAL

ASSOCIATE
'1'

RECLEN EMPNAME (KEY) BRECSEC EMPINSCL EMPVEH (CONTROL FIELD)

HEX
DEC

SUBFILE
'Z'

RECLEN EMPKID (KEY) EMPKIDPK (PARENT KEY) EMPKIDR KIDNAME KIDPET

HEX
DEC

SUBFILE
'Y'

RECLEN EMPVEH (KEY) EMPVENPK (PARENT KEY) EMPVEHR VEHMAKE (CODE)

HEX
DEC

INDEX
'A'

RECLEN EMPAYCL (KEY) EMPNAME (CROSS REFERENCES)

HEXADECIMAL
DECIMAL

INDEX
'B'

RECLEN KIDNAME (KEY) EMPKIDID (CROSS REFERENCES)

HEX
DEC

INDEX
'C'

RECLEN EMPINSCL EMPNAME (CROSS REFERENCES) (KEY)

HEX
DEC

INDEX
'D'

RECLEN VEHMAKE (KEY) EMPVEHID (CROSS REFERENCES) (EXTERNAL FORM)

HEX
DEC

NOT REPRODUCIBLE

HEX DEC
00 0
100 256
200 512
300 768
400 1024
500 1280
600 1536
700 1792
800 2048
900 2304
A00 2560
B00 2816
C00 3072
D00 3328
E00 3584
F00 3840

CHARACTERISTIC CODES
C - character, 8-bit code
X - hexadecimal, 4-bit code
B - binary
F - fixed-point, full word
H - fixed-point, halfword
E - floating-point, full word
P - packed decimal
A - address value, full word
V - address, external symbol

Anchor	Associate	Subfile	Subfile
Region: 'PLEX\$\$ '	'PLEX\$\$1'	'PLEX\$\$Z'	'PLEX\$\$Y'
file descriptor	file descriptor	file descriptor	file descriptor
Field descriptors: RECLN EMPNAME (Key) FILEKEY RECSEC EMPTYPE EMPPAYCL EMPKID (control) EMPKIDID EMPKIDPK EMPKIDRS KIDNAME KIDPET KIDID RECSECL EMPINSCL EMPVEH EMPVEHID EMPVEHPK EMPVEHRS VEHAIRCD VEHMAKE	RECLN EMPNAME (Key) FILEKEY RECSECL EMPINSCL EMPVEH (control) EMPVEHID EMPVEHPK EMPVEHRS VEHAIRCD VEHMAKE	RECLN EMPKIDID (Key) EMPKIDPK EMPKIDRS KIDNAME KIDPET KIDID	RECLN EMPVEHID (Key) EMPVEHPK EMPVEHRS VEHAIRCD VEHMAKE
dummy descriptors			

Index	Index	Index	Index
Region: 'PLEX\$\$A'	'PLEX\$\$B'	'PLEX\$\$C'	'PLEX\$\$D'
file descriptor	file descriptor	file descriptor	file descriptor
Field descriptors: RECLN Key: EMPPAYCL Cross references: EMPNAME	RECLN KIDNAME EMPKIDID	RECLN EMPINSCL EMPNAME	RECLN VEHMAKE EMPVEHID

FIGURE 3. SAMPLE DATAPLEX DESCRIPTOR LIST

REGION	DESCOK	FILETYPE	DESCRCT	BSELNGTH	SPANNED	DATA	MNTNABLE	MNTNING	LOADABLE	RECSECFP
PLEX\$\$	ON	ANCHOR	21	17	OFF	OFF	ON	OFF	ON	14
PLEX\$\$1	ON	ASSOCIATE	11	20	OFF	OFF	ON	OFF	ON	14
PLEX\$\$Z	ON	SUBFILE	7	18	OFF	OFF	ON	OFF	ON	17
PLEX\$\$Y	ON	SUBFILE	6	21	OFF	OFF	ON	OFF	ON	17
PLEX\$\$A	ON	INDEX	3	7	ON	OFF	ON	OFF	ON	null
PLEX\$\$B	ON	INDEX	3	14	OFF	OFF	ON	OFF	ON	null
PLEX\$\$C	ON	INDEX	3	9	OFF	OFF	ON	OFF	ON	null
PLEX\$\$D	ON	INDEX	3	14	OFF	OFF	ON	OFF	ON	null

FIGURE 4. SAMPLE FILE DESCRIPTORS

SAMPLE PRIMARY DIRECT FIELD DESCRIPTORS

FLDNAME	SECURITY	READONLY	VARELT=FIXED/VARYING	BITFLD	FLDPOSIT	FLDLEN	NUMALIGN	ELTLIM	ELTLEN	VARELT=FIXED/VARYING	UNIQUELT	GENERCRT	VALIDRTN	VALIDARG	REFORMAT	INFILE	INDEXEXT
PLEX\$\$ EMPNAME	X	OFF	F	OFF	4	10	OFF					()	()	()	()	X	
PLEX\$\$ RECSEC	()	OFF	F	OFF	14	1	OFF					DBCYTHX	X	X	DBFMTHX		
PLEX\$\$ EMPPAYCL	()	OFF	F	OFF	15	2	OFF					()	()	()	()	A	OFF
PLEX\$\$1EMPNAME	X	OFF	F	OFF	4	10	*					*	*	*	*	X	
PLEX\$\$1RECSECI	()	OFF	F	OFF	14	1	OFF					DBCVTTHX	X	X	DBFMTHX		
PLEX\$\$1EMPINSCL	()	OFF	F	OFF	15	5	OFF					()	()	()	()	C	OFF
PLEX\$\$ZEMPKIDID	X	OFF	F	OFF	4	3	ON					DBCVTID	X	X	DBFMTID	X	
PLEX\$\$ZEMPKIDRS	()	OFF	F	OFF	17	1	OFF					DBCVTTHX	X	X	DBFMTHX		
PLEX\$\$ZKIDNAME	()	OFF	V		1	10	OFF					()	()	()	()	B	OFF
PLEX\$\$ZKIDPET	()	OFF	V		2	40	OFF	5	10	V	OFF	()	()	()	()	()	()
PLEX\$\$YEMPVEHID	X	OFF	F	OFF	4	3	ON					DBCVTID	X	X	DBFMTID	X	
PLEX\$\$YEMPVEHRS	()	OFF	F	OFF	17	1	OFF					DBCVTTHX	X	X	DBFMTHX		
PLEX\$\$YVEHAIRCD	()	OFF	F	ON	18	0	OFF					()	()	()	()	X	
PLEX\$\$YVEHMAKE	()	OFF	F	OFF	19	2	OFF					()	()	()	()	D	ON

SAMPLE SECONDARY DIRECT FIELD DESCRIPTORS

FLDNAME	SECURITY	READONLY	VARELT=FIXED/VARYING	BITFLD	FLDPOSIT	FLDLEN	NUMALIGN	ELTLIM	ELTLEN	VARELT=FIXED/VARYING	UNIQUELT	GENERCRT	VALIDRTN	VALIDARG	REFORMAT	INFILE	INDEXEXT	SUBCNTRL	SUBFILE
PLEX\$\$ RECLN	()	ON	F	OFF	0	4	ON												
PLEX\$\$ EMPKID	()	ON	V		1	4000	ON	20	3	F	ON	DBCVTTRL	X	X	DBFMTRL	X		X	X
												DBCVTID	X	X	DBFMTID	X		ON	Z
PLEX\$\$1RECLN	()	ON	F	OFF	0	4	ON												
PLEX\$\$1EMPVEH	()	ON	V		1	4000	ON	5	3	F	ON	DBCVTTRL	X	X	DBFMTRL	X		X	X
												DBCVTID	X	X	DBFMTID	X		ON	Y
PLEX\$\$ZRECLN	()	ON	F	OFF	0	4	ON												
PLEX\$\$ZEMPKIDPK	()	ON	F	OFF	7	10	*					DBCVTTRL	X	X	DBFMTRL	X		X	X
												*	*	*	*	X			
PLEX\$\$YRECLN	()	ON	F	OFF	0	4	ON												
PLEX\$\$YEMPVEHPK	()	ON	F	OFF	7	10	*					DBCVTTRL	X	X	DBFMTRL	X		X	X
												*	*	*	*	X			

FIGURE 5. SAMPLE DIRECT FIELD DESCRIPTORS

FLDNAME	SECURITY	READONLY	NAMEFLD	REFORMAT
PLEX\$\$ FILEKEY	X	ON	(EMPNAME)	X
PLEX\$\$ EMPTYTYPE	()	ON	(EMPPAYCL,EMPINSCL)	()
PLEX\$\$1FILEKEY	X	ON	(EMPNAME)	X
PLEX\$\$ZKIDID	()	ON	(KIDNAME,EMPKIDPK)	()

FIGURE 6. SAMPLE INDIRECT SECONDARY FIELD DESCRIPTORS

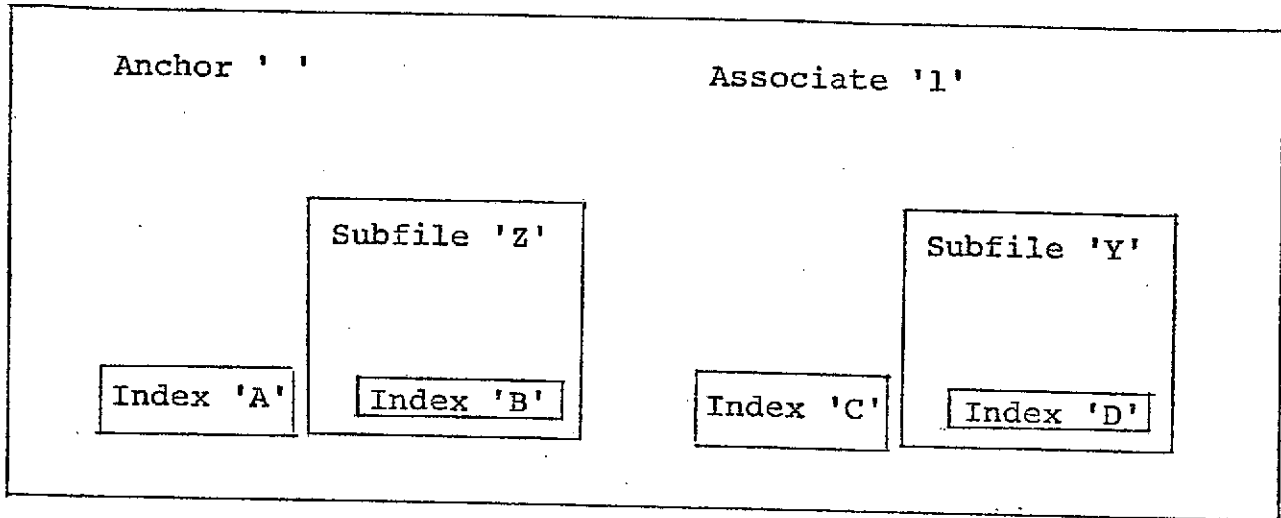


FIGURE 7. SAMPLE DESCRIPTOR FILE

	FLDNAME	ASSOCFIL	SUBFILE	SUBCNTRL	INVFILE
PLEX\$\$	EMPKIDID				
"	EMPKIDPK		Z	OFF	
"	EMPKIDRS		Z	OFF	
"	KIDNAME		Z	OFF	
"	KIDPET		Z	OFF	
"	KIDID		Z	OFF	B
"	RECSECI		Z	OFF	
"	EMPINSCL	1			
"	EMPVEH	1			
"	EMPVEHID	1	Y	ON	C
"	EMPVEHPK	1	Y	OFF	
"	EMPVEHRS	1	Y	OFF	
"	VEHAIRCD	1	Y	OFF	
"	VEHMAKE	1	Y	OFF	
PLEX\$\$1	EMPVEHID		Y	OFF	
"	EMPVEHPK		Y	OFF	D
"	EMPVEHRS		Y	OFF	
"	VEHAIRCD		Y	OFF	
"	VEHMAKE		Y	OFF	D

FIGURE 8. SAMPLE DUMMY FIELD DESCRIPTORS

TOPIC B.8 - DATA BASE EXECUTIVE

A. DATA SET NAME:

DBPL/I - DBLIST Interface

B. CREATED BY:

DB Preprocessor Function

C. TYPE OF FILE:

(4) Table

D. ORGANIZATION:

Documentary Table

E. KEY IDENTIFIER (CONTROL FIELD):

Not Applicable

F. RECORD LENGTH:

Not Applicable

G. BLOCKING FACTOR:

Not Applicable

H. PURPOSE:

The DBPL/I - DBLIST Interface (see Table 1) specifies the DBLIST entry point name and the argument types and order for the various DBPL/I statements. Thus, it serves to specify for the DB Preprocessor function (see Section IV, Topic B.5) what CALL statements are to be generated for each DBPL/I statement. Conversely, it specifies for DBLIST (see Section IV, Topic B.5) what entry points will be entered and what and how information will be available at execution time for the performance of the various statement actions.

The various entry points and their argument types are declared by source code in the macro library member DBTEXT. Any program that includes the DB preprocessor also is given DBTEXT by an INCLUDE statement in DB.

TABLE 1.

DBFL/I	GENERATED PL/I
FREE LIST;	CALL EBPAC;
FRFE LIST(p1,p2);	CALL DBPACP(p1);
GET LIST(p1) KEY(0);	CALL DBGLK0(p1);
GET LIST(p1) KEY INTO (st);	CALL DBGLKI(p1,st);
GET LIST(p1) INTERNAL KEY INTO (st);	CALL DBGLIK(p1,st);
GET LIST(p1) KEY(n) INTO(st);	CALL DBGLKN(p1,n,st);
GET LIST(p1) KFY SET(p2);	CALL DBGLKS(p1,p2);
SET LIST(p2) SIZE(n) LIKE IIST(p1);	CALL LESLLL(p2,n,p1);

where:

p1,p2 are POINTER
n is FIXED BINARY(31)
st is CHARACTER(-) VARYING

TOEIC B.9 - SETS INFORMATION FILE

A. DATA SET NAME:

SETSINFO - Set-File Information File

B. CREATED BY:

DBSETI - Utility Module

C. TYPE OF FILE:

Non-Data Base File

D. ORGANIZATION:

Sequential

E. KEY IDENTIFIER:

Not Applicable

F. RECORD LENGTH:

Fixed, 100 bytes

G. BLOCKING FACTOR:

Unblocked

H. PURPOSE:

This file describes the attributes of the Set File; it is created at the time the Set File is pre-formatted. The Information File contains the number of devices allocated to the Set File, the number of records per device, and the record size. This file is read at NASIS initialization by MTTSTART and the information is posted into the MASTAB table for DBCSET.

TOPIC B.10 - SET FILE

A. DATA SET NAME:

SETFILE - Set File

B. CREATED BY:

SETINIT - Utility Module

C. TYPE OF FILE:

Non-Data Base File

D. ORGANIZATION:

Random (BEAM)

E. KEY IDENTIFIER:

Not Applicable

F. RECORD LENGTH:

Fixed (actual length parameterized at creation)

G. BLOCKING FACTOR:

Unblocked

H. PURPOSE:

This file is created and pre-formatted by SETINIT, which is attached by the DBSETI stand-alone utility. Its record length and size is optional. The characteristics of this file reside in the SETSINFO File upon creation. Once pre-formatted, this file is used by the NASIS Set Manager, DEOSET, for storing sets.

TOPIC C.1 - UTILITIES

A. DATA SET NAME:

NASIS USERIDS

B. CREATED BY:

DBJOIN

C. TYPE OF FILE:

ISAM

D. ORGANIZATION:

Variable format

E. KEY IDENTIFIER (CONTROL FIELD):

8 Character NASISID, key length 8, key position 5.

F. RECORD LENGTH:

4000 bytes

G. BLOCKING FACTOR:

Not Applicable

H. PURPOSE:

Maintain for each NASIS ID, the corresponding password, timeslice, user authority and list of valid files.

TOEIC D.1 - MAINTENANCE

A. DATA SET NAME:

NDBLOAD ERROR_CODES Table

B. CREATED BY:

NDBLOAD

C. TYPE OF FILE:

Core table

D. ORGANIZATION:

Sequential array

E. KEY IDENTIFIER (CONTROL FIELD):

Not Applicable

F. RECORD LENGTH:

Defined as (0:216) character (1)

G. BLOCKING FACTOR:

Not Applicable

H. PURPOSE:

This table contains codes to be used to control the action taken for each LEPAC error that may occur.

TOEIC D.2 - MAINTENANCE

A. DATA SET NAME:

TRNSCT Data Set Descriptors

B. CREATED BY:

CORRECT (NEBCCBR)

C. TYPE OF FILE:

QISAM - Anchor

D. ORGANIZATION:

Indexed Sequential

E. KEY IDENTIFIER (CONTROL FIELD):

Offset of 5, fixed field (255)

F. RECORD LENGTH:

Varying (4001)

G. BLOCKING FACTOR:

Not Applicable

H. PURPOSE:

The purpose of the transaction data base is to serve as a temporary repository for changes to one of the data base datafiles, until the file owner can validate the change and execute the maintenance program to apply it.

The TRNSCT file will consist of fields defined as follows:

KEY - This will be an alphanumeric fixed field 255 bytes in length consisting of the file name (padded to six characters with \$) concatenated with the OWNER-ID, (padded to eight characters with *) of the affected file concatenated with the anchor's key. The last fourteen (14) bytes of the key will be the time stamp field.

NASISID - This is an alphanumeric fixed field, 8 bytes in length, containing the NASIS-ID of where the transaction was created.

CPCODE - This is an alphanumeric fixed field 3 bytes in length, which indicates the operation to be performed.

FIELD - This is an alphanumeric fixed field, 8 bytes in length, which indicates the field of a file which is to be updated. START - This is an alphanumeric fixed field, 4 bytes in length. For field context operation, this field will contain the starting location of the context.

END - This is an alphanumeric fixed field, 4 bytes in length. For field context operation, this field will contain the ending location of the context.

SUBKEY - This is an alphanumeric fixed field, 10 bytes in length, which indicates the subfile key to be corrected.

SUBCTL - This is an alphanumeric fixed field, 8 bytes in length, which is the subfile control field for subfile being corrected.

OLDDATA - This is a varying alphanumeric field, maximum 3694 bytes long. This will be the old data field to change.

NEWDATA - This is a varying alphanumeric field, maximum 3694 bytes long. This will be the new data field to be added or updated.

There are many advantages to having the transactions on a data base. One important consideration is that conversion, validation and reformatting routine can be written and used to check the data as it enters the data base, thus causing many errors to be detected before maintenance is ever run.

There are many advantages to having the transactions on a data base. One important consideration is that conversion, validation and reformatting routines can be written and used to check the data as it enters the data base, thus causing many errors to be detected before maintenance is ever run.

TOPIC D.3 - MAINTENANCE

A. DATA SET NAME:

CORRECT Data Display Format

B. CREATED BY:

CORRECT (NDBCORR)

C. TYPE OF FILE:

Terminal Display

D. ORGANIZATION:

Not Applicable

E. KEY IDENTIFIER (CONTROL FIELD):

Not Applicable

F. RECORD LENGTH:

Not Applicable

G. BLOCKING FACTOR:

Not Applicable

H. PURPOSE:

The purpose of this display is to present as much of the data contained in the field as possible to the user. By doing so, in a reasonable format, the user can examine the data and make any required adjustments to correct errors that he detects.

CORRECT XXXXXXXX, XXXXXXXXXXXXXXXX

(LENGTH = 80, ELEMENTS = 4)

E001 :XXXXXXXX
E002 :XX
002 :XXXXXX
E003 :XXXXXXXX
E004 :XXXXXXXX

TOEIC D.4 - MAINTENANCE

A. DATA SET NAME:

NDBLOAD Error Data Set 'DBLOAD.ERROR'

B. CREATED BY:

NDBLOAD

C. TYPE OF FILE:

QISAM

D. ORGANIZATION:

Indexed Sequential

E. KEY IDENTIFIER (CONTROL FIELD):

(Must be the same as that of the NDBLOAD input data set).

F. RECORD LENGTH:

(Must be the same as that of the NDBLOAD input data set).

G. BLOCKING FACTOR:

Not Applicable

H. PURPOSE:

This data set serves as the repository for all input records that cannot be successfully loaded.

TOEIC D.5 - MAINTENANCE

A. DATA SET NAME:

Data Base Inverted Index Format

B. CREATED BY:

A descriptor region is created by DBEDIT (the descriptor editor) for each inverted index. It generally consists of a header record and three field descriptor records.

Inverted index records are originally written by either the invert maintenance program or NDBPAC (inverting concurrently with data base loading).

Inverted index records are automatically maintained during data base maintenance by DBPAC.

C. TYPE OF FILE:

(6) NASIS File

D. ORGANIZATION:

QISAM - Queued Indexed Sequential Access Method

E. KEY IDENTIFIER (CONTROL FIELD):

The index key field name is the same as the indexed field name, e.g. AUTHOR, SUBJTERM, KEYWORD etc. The QISAM key length is the maximum length of the indexed values (plus 1 for a spanned index) and may not exceed 255 bytes.

F. RECORD LENGTH:

Variable - 4000 bytes maximum.

G. BLOCKING FACTOR:

Not Applicable

H. PURPOSE:

The purpose of the inverted index files is to give the system a fast, efficient method of storing and accessing the list of records in a data base file that contain a particular data element value.

The records of inverted files consist of a universal

record form with a specialized structure. The structure consists of the concatenation of the following fields:

the QISAM record length field (RECLEN)

4 bytes, fixed length, binary

the QISAM delete byte, 1 byte, fixed length, binary

the QISAM key field

1-254 bytes, fixed length, indexed field element value

optionally suffixed by

1 byte, fixed length, binary record number within region if the index is SPANNED.

the crcss references field

2 bytes, fixed length, binary field length

followed by one or more fixed length

anchor or subfile key values in ascending collating sequence.

In a SPANNED index, the first record of a region has key suffix zero, and possible continuation records (up to 255) in a continuous 'region'. All records in a region, except possibly the last, are maximum length (have the maximum number of whole cross references) after index loading or maintenance with the cross reference values ascending from record to record as well as across each record.

TOEIC D.6 - MAINTENANCE

A. DATA SET NAME:

Descriptor Editor Data Display Format

B. CREATED BY:

Descriptor Editor DISPLAY Command (DBEDDP)

C. TYPE OF FILE:

Terminal Display

D. ORGANIZATION:

Not Applicable

E. KEY IDENTIFIER (CONTROL FIELD):

Not Applicable

F. RECORD LENGTH:

Not Applicable

G. BLOCKING FACTOR:

Not Applicable

H. PURPOSE:

The purpose of this display is to allow the user of the Descriptor Editor to review the specifications for a particular field descriptor.

I. SAMPLE DISPLAY:

```

FIELD NAME.....XXXXXXXXX
FIELD TYPE.....XX
ALIGNMENT.....X
FIELD FORMAT.....XX
FIELD LENGTH.....XXXX
ELEMENT LENGTH.....XXX
ELEMENT NUMBER.....XXXX
UNIQUE ELEMENTS.....X
CONVERSION ROUTINE.....XXXXXXXXX
FORMATTING ROUTINE.....XXXXXXXXX
VALIDATION ROUTINE.....XXXXXXXXX
VALIDATION ARGUMENT.....XXXXXXXXXXXXXXXXXX
XXXXXXXXXXXXXXXXXXXXXXXXXXXXXXXXXXXX
XXXXXXXXXXXXXXXXXXXXXXXXXXXXXXXXXXXX

```

[illegible]

TOEIC D.7 - MAINTENANCE

A. DATA SET NAME:

Descriptor Editor Field Name Display Format

B. CREATED BY;

Descriptor Editor (FIELDS) Command (DBEDFD)

C. TYPE OF FILE:

Terminal Display

D. ORGANIZATION:

Not Applicable

E. KEY IDENTIFIER (CONTROL FIELD):

Not Applicable

F. RECORD LENGTH:

Not Applicable

H. PURPOSE:

The purpose of this display is to allow the user of the Descriptor Editor to review the names of all of the fields described thus far in CREATE mode. In UPDATE mode the user is presented a list of the descriptor descriptor field names.

I. SAMPLE DISPLAY:

```

XXXXXXXXXXXXXXXXXXXXXXXXX
XXXXXXXXX  XXXXXXXX  XXXXXXXX
XXXXXXXXX  XXXXXXXX  XXXXXXXX
XXXXXXXXX  XXXXXXXX

```

NOTE: A Total of 57 names can be displayed on one screen.

TOPIC D.8 - MAINTENANCE

A. DATA SET NAME:

NDBLOAD Input Data Set

B. CREATED BY:

Not Applicable

C. TYPE OF FILE:

QISAM

D. ORGANIZATION:

Sequential

E. KEY IDENTIFIER (CONTROL FIELD):

Usually same as that of the data base to be loaded.

F. RECORD LENGTH:

Not Applicable

G. BLOCKING FACTOR:

Not Applicable

H. PURPOSE:

This data set serves as the source of input records for NDBLOAD.

TOPIC D.9 - MAINTENANCE

A. DATA SET NAME:

Descriptor Editor Listing Format

B. CREATED BY:

Descriptor Editor Print Command (DBEDPR)

C. TYPE OF FILE:

1403 Printer Display

D. ORGANIZATION:

Not Applicable

E. KEY IDENTIFIER (CONTROL FIELD):

Not Applicable

F. RECORD LENGTH:

133

G. BLOCKING FACTOR:

Not Applicable

H. PURPOSE:

The purpose of this display is to list the contents of each descriptor field and each file descriptor in character form.

I. SAMPLE DISPLAY:

See Figure 1

[illegible]

FIGURE 1. SAMPLE LISTING FORMAT

TOPIC D.10 - MAINTENANCE

A. DATA SET NAME:

INVERT Restart File

'INVERT.PARM.'||FILENAME -
where FILENAME is the six character data base name.

B. CREATED BY:

NDBIVRT1 module.

C. TYPE OF FILE:

Sequential

D. ORGANIZATION:

SAM

E. KEY IDENTIFIER (CONTROL FIELD):

Not Applicable

F. RECORD LENGTH:

255 bytes (Variable)

G. BLOCKING FACTOR:

Not Applicable

H. PURPOSE:

This file provides a restart key for the first phase of
INVERT.

TCEIC D.11 - MAINTENANCE

A. DATA SET NAME:

INVERT SORTIN File

'SORTIN.'||FILENAME||'.'||FIELD

1. FILENAME is the six character data base name.
2. FIELD is the 1-8 character field name that is being inverted.

B. CREATED BY:

First step of DBIVRT1.

C. TYPE OF FILE:

Sequential

D. ORGANIZATION:

SAM

E. KEY IDENTIFIER (CONTROL FIELD):

First field is the maximum length value of the field being inverted.

F. RECORD LENGTH:

4000 bytes (Variable). Record consists of maximum length value of field being inverted concatenated with file Key.

G. BLOCKING FACTOR:

Not Applicable

H. PURPOSE:

This file is the input to the second phase of invert, a DSORT utility.

TOPIC D.12 - MAINTENANCE

A. DATA SET NAME:

INVERT SORTOUT File

'SORTOUT.'||FILENAME||'.'||FIELD

1. FILENAME is the six character data base name.
2. FIELD is the 1-8 character field name that is being inverted.

B. CREATED BY:

Sort step of DBSIVRT.

C. TYPE OF FILE:

Sequential

D. ORGANIZATION:

QSAM

E. KEY IDENTIFIER (CONTROL FIELD):

First field is the maximum length value of the field being inverted.

F. RECORD LENGTH:

4000 bytes (Variable). Record consists of maximum length value of field being inverted concatenated with the file Key.

G. BLOCKING FACTOR:

Not Applicable

H. PURPOSE:

This file is the input to DBIVRT2.

TCEIC D.13 - MAINTENANCE

A. DATA SET NAME:

INVERT PLEX File

'PIEX.'||FILENAME||'.'||FIELD

1. FILENAME is the six character data base name.
2. FIELD is the 1-8 character field name that is being inverted.

B. CREATED BY:

Step three of DBIVRT2.

C. TYPE OF FILE:

Indexed Sequential

D. ORGANIZATION:

ISAM

E. KEY IDENTIFIER (CONTROL FIELD):

Key of file is internal field value being inverted concatenated with blanks up to the maximum external field length. If index file is spanned, span character is concatenated as last position of Key.

F. RECORD LENGTH:

4000 bytes (Variable). Record is identical in format as an index file record.

G. BLOCKING FACTOR:

Not Applicable

H. PURPOSE:

This file is the input to the last step, translation step, of DBIVRT2.

TOPIC D.14 - MAINTENANCE

A. DATA SET NAME:

INVERT RANGE File

'RANGE.'||FILENAME||'.'||FIELD

1. FILENAME is the six character data base name.
2. FIELD is the 1-8 character field name that is being inverted.

B. CREATED BY:

DBIVRT2

C. TYPE OF FILE:

Indexed Sequential

D. ORGANIZATION:

ISAM

E. KEY IDENTIFIER (CONTROL FIELD):

Key of the file is the maximum length value of the field being inverted. If index file is spanned, span character is concatenated as last position of Key.

F. RECORD LENGTH:

4000 bytes (Variable). Record is identical to index file record.

G. BLOCKING FACTOR:

Not Applicable

H. PURPOSE:

This file is the input to the index merge module for index updates.

TOPIC D.15 - MAINTENANCE

A. DATA SET NAME:

DESCRP.CHKPOINT

B. CREATED BY:

Descriptor Editor Checkpoint (DBEDCP)

C. TYPE OF FILE:

TSS VAM

D. ORGANIZATION:

Sequential

E. KEY IDENTIFIER (CONTROL FIELD):

Not Applicable

F. RECORD LENGTH:

The records are of varying length of which the maximum is dynamically determined at execution time. The maximum possible value is 4000.

G. BLOCKING FACTOR:

Not Applicable

H. PURPOSE:

The purpose of this dataset is for storing sufficient information from the descriptor tables so that the user can continue creating the descriptor file at a future time through use of the RESTORE command.

The first record consists of those items from the X structure whose value must be preserved. The second record consists of the entire content of the FIELD structure. The next group of records will contain the field descriptor information. There will be one record for each existing field, consisting of the information in the appropriate FLD structure concatenated with the information contained in the appropriate SECURITY, SUPER, and VALID structures where applicable. Following the field descriptor records are records containing the header descriptor information, one for each existing file. These records consist of the information from the appropriate HDR structure

concatenated to the information from the proper RECSEC structure when applicable.

TOEIC D.16 - MAINTENANCE

A. DATA SET NAME:

'INDMRG.PARM.'||FILENAME - FILENAME is the six character data base name.

B. CREATED BY:

NDBINDM2 (index merge module)

C. TYPE OF FILE:

SEQUENTIAL

D. ORGANIZATION:

QSAM

E. KEY IDENTIFIER (CONTROL FIELD):

Not Applicable

F. RECORD LENGTH:

255 Bytes (variable)

G. BLOCKING FACTOR:

Not Applicable

H. PURPOSE:

This file provides a restart key for restart of IBINDM2.

TCEIC D.17 - MAINTENANCE

A. DATA SET NAME:

Descriptor Editor REVIEW Display Format

B. CREATED BY:

Descriptor Editor REVIEW Command (DBEDRV)

C. TYPE OF FILE:

Terminal Display

D. ORGANIZATION:

Not Applicable

E. KEY IDENTIFIER (CONTROL FIELD):

Not Applicable

F. RECORD LENGTH:

Not Applicable

G. BLOCKING FACTOR:

Not Applicable

H. PURPOSE:

The purpose of this display is to allow the user of the Descriptor Editor to review the exact contents of any descriptor record in any descriptor region.

I. SAMPLE DISPLAYS:

FILE DESCRIPTOR

[illegible]

FIELD DESCRIPTOR

```

-----
FLDNAME =XXXXXXXXX  ASSOCFIL=X
SUBFILE =X          INVFILE =X
READONLY=X          INFILE  =X
VARFLD  =X          BITFLD  =X
NUMALIGN=X          VAPFLT  =X
UNIQUELT=X          INDEXEXT=X
GENERCPT=XXXXXXXXX  VALIDRTN=XXXXXXXXX
REFORMAT=XXXXXXXXXX FLDPCSIT=XXXXX
FLDEN    =XXXXXX    DFLEN    =XXXXXX
FLDLN    =XXXXXX    DELTIM   =XXXXXX
ELTIM    =XXXXXX    DELTIEN  =XXXXXX
SPARE    =XXXXXXXXXXXXXXXXXXXX
VALIDARG=XXXXXXXXXXXXXXXXXXXXXXXXXXXXXXXXXXXX
          XXXXXXXXXXXXXXXXXXXXXXXXXXXXXXXXXXXX
          XXXXXXXXXXXXXXXXXXXXXXXXXXXXXXXXXXXX
          XXXXXXXX
NAMEFLD  =X.XXXXXXXX  X.XXXXXXXX
          X.XXXXXXXX  X.XXXXXXXX
          X.XXXXXXXX  X.XXXXXXXX
          X.XXXXXXXX  X.XXXXXXXX
          X.XXXXXXXX  X.XXXXXXXX
          X.XXXXXXXX  X.XXXXXXXX
          X.XXXXXXXX  X.XXXXXXXX
          X.XXXXXXXX  X.XXXXXXXX
SECURITY=XXXXXXXXXX  XXXXXXXXX  XXXXXXXXX
          XXXXXXXXX  XXXXXXXXX  XXXXXXXXX
          XXXXXXXXX  XXXXXXXXX  XXXXXXXXX
          XXXXXXXXX  XXXXXXXXX  XXXXXXXXX
          XXXXXXXXX  XXXXXXXXX  XXXXXXXXX
          XXXXXXXXX  XXXXXXXXX  XXXXXXXXX

```

TOPIC D.18 - MAINTENANCE

A. DATA SET NAME:

DEFIELD which consists of the external structure FIELD

B. CREATED BY:

Not Applicable

C. TYPE OF FILE:

Table

D. ORGANIZATION:

PL/I Data Structure

E. KEY IDENTIFIER (CONTROL FIELD):

Not Applicable

F. RECORD LENGTH:

Not Applicable

G. BLOCKING FACTOR:

Not Applicable

H. PURPOSE:

This table is used to contain the names and core locations of all the field descriptor during the running of a retrieval session.

I. PL/I DECLARATION:

THIS STRUCTURE IS USED TO CONTAIN THE FIELD NAMES AND THEIR RESPECTIVE FLD PCINTERS.

```

1 FIELD      BASED (X.FIELD_PTR), /* FIELD NAMES AND */
              /* PCINTERS STRUCTURE. */
3 RECLEN     BIN (31) FIXED, /* RECORD LENGTH FOR */
              /* ASMPUT. THIS IS USED IN */
              /* CHKPOINT COMMAND. IT IS */
              /* SET EQUAL TO ELT LENGTH OF */
              /* FIELD STRUCTURE. */
3 LAST      BIN FIXED, /* INDEX OF IAST TABLE ENTRY. */
3#          BIN FIXED, /* NUMBER OF ENTRIES IN TABLE.*/
3 A (X.#FN REFER (FIELD.*)),
5 NAME CHAR (8), /* FIELDNAME ARRAY. */
5 PTR PTR;      /* FLD STRUCTURE POINTERS. */

```

TOPIC D.19 - MAINTENANCE

A. DATA SET NAME:

DERECSEC which consists of the structures RECSEC and RECSEC STR

B. CREATED BY:

Not Applicable

C. TYPE OF FILE:

Table

D. ORGANIZATION:

PL/I Data Structure

E. KEY IDENTIFIER (CONTROL FIELD):

Not Applicable

F. RECORD LENGTH:

Not Applicable

G. BLOCKING FACTOR:

Not Applicable

H. PURPOSE:

The RECSEC structure is used to contain the record security codes that pertain to a given file. RECSEC_STR is a character string overlay of the RECSEC structure.

I. PL/I DECLARATION:

THE RECSEC STRUCTURE IS USED TO STORE THE RECORD SECURITY CODES AND SECURITY MASKS USED TO DETERMINE RECORD SECURITY. THE RECSEC STRUCTURE IS POINTED TO BE EDR.RSECTYCD FIELD WHEN THE FILE HAS RECORD SECURITY DEFEND ON IT.

```

1 RECSEC      BASED (X.RSEC_PTR), /* RECORD SECURITY  */
               /* CODES STRUCTURE.                */
3 #           BIN FIXED, /* NUMBER OF SECURITY CODES. */
3 SECURITY (18),
    5 CODES CHAR (8), /* USER PASSWORD. */
    5 MASK CHAR (2), /* RECORD ACCESS CODE. */
3 CHANGED (18) BIT (1), /* ONE FLAG FOR EACH SECURITY */
                       /* CODE. IF ON THEN REPUT NEW */
                       /* VALUE. */
3 FILLER CHAR (8); /* NEEDED FOR PLI BUG. */

```

THIS STRUCTURE IS A CHARACTER STRING OVERLAY OF THE RECSEC STRUCTURE. IT IS USED FOR MAKING COPIES OF THE RECSEC STRUCTURE.

```

DCL RECSEC_STR CHAR (193) BASED (X.RSEC_PTR);
               /* RECSEC STRUCTURE OVERLAY. */

```

TOPIC D.20 - MAINTENANCE

A. DATA SET NAME:

DESECUB which consists of the structure SECURITY and SECURITY_STR.

B. CREATED BY:

Not Applicable

C. TYPE OF FILE:

Table

D. ORGANIZATION:

PL/I Data Structure

E. KEY IDENTIFIER (CONTROL FIELD):

Not Applicable

F. RECORD LENGTH:

Not Applicable

G. BLOCKING FACTOR:

Not Applicable

H. PURPOSE:

The SECURITY structure is used to contain the security codes defining field security for a given field. SECURITY_STR is a character string overlay of the SECURITY structure.

I. PL/I DECLARATION

THE SECURITY STRUCTURE IS USED TO STORE THE FIELD SECURITY CODES DEFINED FOR A GIVEN FIELD. IT IS POINTED TO BY THE FLD.SECURITY FIELD ON WHICH THIS FIELD SECURITY IS DEFINED.

```
1 SECURITY      BASED (X.FSEC_PTR), /* FIELD SECURITY      */
                /* CODES STRUCTURE.          */
3 #            BIN FIXED, /* NUMBER OF SECURITY CCDES */
                /* FOR THIS FIELD.          */
3 CODE (18) CHAR (8), /* SECURITY CODE VALUES.    */
3 CHANGED (18) BIT (1), /* ONE FLAG FOR EACH SECURITY */
                /* CODE. IF ON THEN REPUT THE */
                /* NEW VALUE.                */
3 FILLER       CHAR (8); /* NEEDED FOR PLI BUG.      */
```

THIS STRUCTURE IS A CHARACTER STRING OVERLAY OF THE SECURITY STRUCTURE. IT IS USED FOR MAKING COPIES OF THE SECURITY STRUCTURE.

```
DCL SECURITY_STR CHAR (157) BASED (X.FSEC_PTR);
                /* SECURITY STRUCTURE OVERLAY.*/
```

TOEIC D.21 - MAINTENANCE

A. DATA SET NAME:

DESUPER which consists of the structure SUPER and SUPER_STR.

B. CREATED BY:

Not Applicable

C. TYPE OF FILE:

Table

D. ORGANIZATION:

FL/I Data Structure.

E. KEY IDENTIFIER (CONTROL FIELD):

Not Applicable

F. RECORD LENGTH:

Not Applicable

G. BLOCKING FACTOR:

Not Applicable

H. PURPOSE:

The SUPER structure is used to contain the superfield component information of a field descriptor. SUPER_STR is a character string overlay of the SUPER structure.

I. FL/I DECLARATION:

THE SUPER STRUCTURE IS USED TO STORE THE SUPERFIELD COMPONENTS OF A SUPER DESCRIPTOR. IT IS POINTED TO BY THE PLD.NAMEFLD OF THE DEFINING SUPERFIELD.

```

1 SUPER          BASED (X.SUPER_PTR), /* SUPER FIELD          */
                                     /* COMPONENT FIELDNAMES  */
                                     /* STRUCTURE.           */
3 #              BIN FIXED, /* NUMBER OF COMPONENT NAMES. */
3 NAME (16),
    5 CODE CHAR (1), /* INTERNAL-EXTERNAL INDICATOR*/
    5 FIELD CHAR(8), /* COMPONENT FIELD NAMES.     */
3 CHANGED (16) BIT (1), /* ONE FLAG FOR EACH COMPONENT*/
                                     /* NAME. IF ON THEN REPUT THIS*/
                                     /* COMPONENT NAME.           */
3 FILLER         CHAR (8); /* NEEDED FOR PLI BUG.       */

```

THIS STRUCTURE IS A CHARACTER STRING OVERLAY OF THE
SUPER STRUCTURE. IT IS USED FOR MAKING COPIES OF THE
SUPER STRUCTURE.

```
DCL SUPER_STR      CHAR (156) BASED (X.SUPER_PTR);
/* SUPER STRUCTURE OVERLAY.  */
```

TOPIC D.22 - MAINTENANCE

A. DATA SET NAME:

DEVALID which consists of the structure VALID

B. CREATED BY:

Not Applicable

C. TYPE OF FILE:

Table

D. ORGANIZATION:

PL/I Data Structure

E. KEY IDENTIFIER (CONTROL FIELD):

Not Applicable

F. RECORD LENGTH:

Not Applicable

G. BLOCKING FACTOR:

Not Applicable

H. PURPOSE:

This structure is used to contain the validation argument for a field descriptor.

I. PL/I DECLARATION:

THE VALID STRUCTURE IS USED TO STORE A VALIDATION ARGUMENT IF ONE IS DEFINED FOR THE FIELD. IT IS POINTED TO BY FLD.VALIDARG IN THE FIELD TO WHICH THIS ARGUMENT BEICNGS.

```

1 VALID      BASED (X.ARG_FTR), /* VALIDATION ARGUMENT */
              /*      STRUCTURE.          */
3 LENGTH     BIN FIXED, /* LENGTH OF VALIDATION */
              /*      ARGUMENT.            */
3 ARGUMENT   CHAR (X.LVA REFER (VALID.LENGTH));
              /* VALIDATION ARGUMENT.      */

```

TCEIC D.23 - MAINTENANCE

A. DATA SET NAME:

DEFLD which consists of the based structures

FLD and FLD STRING

B. CREATED BY:

Not Applicable

C. TYPE OF FILE:

Table

D. ORGANIZATION:

PL/I Data Structure

E. KEY IDENTIFIER (CONTROL FIELD):

Not Applicable

F. RECORD LENGTH

Not Applicable

G. BLOCKING FACTOR:

Not Applicable

H. PURPOSE:

The FLD structure is used to contain the information describing a field descriptor. FLD_STRING is a character string used to overlay the FLD structure.

I. PL/I DECLARATION:

THIS STRUCTURE IS USED TO STORE THE INFORMATION
DEFINING A FIELD DESCRIPTOR.

```

1 FLD          BASED (X.FLD PTR), /* FIELD DESCRIPTOR */
                /* STRUCTURE. */
3 BACKWARD PTR, /* BACKWARD FID POINTER. */
3 FORWARD PTR, /* FORWARD FLD POINTER. */
3 FLDNAME CHAR (8), /* FIELD NAME. */
3 ASSOCFIL CHAR (1), /* ASSOCIATE FILE IDENTIFIER. */
3 SUBFILE CHAR (1), /* SUBFILE IDENTIFIER. */
3 INVFILE CHAR (1), /* INVERTED FILE IDENTIFIER. */
3 READONLY CHAR (1), /* FIELD READ ONLY FLAG. */
3 SUBCNTRL CHAR (1), /* FIELD A SUBFILE CONTROL FLD*/
3 VARFLD CHAR (1), /* VARYING LENGTH FIELD FLAG. */
3 BITFLD CHAR (1), /* FIELD IS FIXED LENGTH */
                /* BIT STRING OF LENGTH ONE. */
3 NUMALIGN CHAR (1), /* FIELD ALIGNMENT FLAG. */
3 VARELT CHAR (1), /* FIELD ELEMENTS OF VARYING */
                /* LENGTH FLAG. */
3 UNIQUELT CHAR (1), /* ELEMENTS UNIQUE FLAG. */
3 INDEXEXT CHAR (1), /* INDEX KEYS TO BE IN */
                /* INTERNAL OR EXTERNAL FORM */
                /* FLAG. */
3 FILLER CHAR (1), /* BOUNDARY ALIGNMENT. */
3 GENERCRT CHAR (8), /* CONVERSION ROUTINE NAME. */
3 VALIDRTN CHAR (8), /* VALIDATION ROUTINE NAME. */
3 REFORMAT CHAR (8), /* FORMATTING ROUTINE NAME. */
3 SPARE CHAR (16), /* UNUSED DESCRIPTOR FIELD. */
3 FLDPCIT BIN FIXED, /* FIELD POSITION VALUE. */
3 FLDLEN BIN FIXED, /* FIELD LENGTH VALUE. */
3 DFLDIEN BIN FIXED, /* MAXIMUM FIELD LENGTH OF ALL*/
                /* VALUES STORED ON THE DATA */
                /* BASE. */
3 ELTLIM BIN FIXED, /* MAX NUMBER OF ELEMENTS/FLD.*/
3 DELTIM BIN FIXED, /* MAXIMUM ELEMENTS STORED IN */
                /* THIS FIELD IN THE DATA BASE*/
3 ELTLEN BIN FIXED, /* ELEMENT LENGTH VALUE. */
3 DELTLEN BIN FIXED, /* MAXIMUM ELEMENT LENGTH */
                /* OF ALL OF THE ELEMENTS */
                /* STORED FOR THIS FIELD IN */
                /* THE DATA BASE. */
3 VALIDARG PTR, /* POINTER TO VALIDATION */
                /* ARGUMENT IF ANY. */
3 NAMEFLD PTR, /* POINTER TO LIST OF FIELD */
                /* NAMES MAKING UP SUPER FIELD*/
3 SECURITY PTR, /* POINTER TO FIELD SECURITY */
                /* CODES IF ANY. */
3 BASEFLD CHAR (8), /* THE FIELDNAME ON WHICH A */
                /* SUBFIELD IS TO BE DEFINED. */
3 OFFSET BIN FIXED, /* THE OFFSET WITHIN THE BASE */
                /* FIELD THAT THE SUBFIELD */

```

```

/*      STARTS.      */
3 FILE_LIST BIN FIXED, /* ON WHICH ENTRY IN FILE_TAB */
/* HAS THIS FIELD BEEN HUNG. */
3 FLDTYPE    BIN FIXED, /* ENTRY INTO FIELD TYPE TABLE*/
/* DEFINING WHICH TYPE OF    */
/* FIELD THIS IS.            */
3 CHANGED (28) BIT (1), /* ONE FLAG FOR EACH ITEM IN */
/* FLD STRUCTURE. IF ON THEN */
/* PUT NEW VALUE IN DESCRIPTOR*/
/*      FILE.      */
3 FILLER2      CHAR (8); /* NEEDED FOR PL/I BUG.      */

```

THIS STRUCTURE IS A CHARACTER STRING OVERLAY ON THE FLD STRUCTURE. IT IS USED FOR MAKING COPIES OF THE FLD STRUCTURE.

```

DCI FLD_STRING      CHAR (122) BASED (X.FLD_PTR);
/* FLD STRUCTURE OVERLAY.      */

```

TCHIC D.24 - MAINTENANCE

A. DATA SET NAME:

DEXINIT which consists of the X external data structure including all initialization values.

B. CREATED BY:

Not Applicable

C. TYPE OF FILE:

Table

D. ORGANIZATION:

PL/I Data Structure

E. KEY IDENTIFIER (CONTROL FIELD):

Not Applicable

F. RECORD LENGTH:

Not Applicable

G. BLOCKING FACTOR:

Not Applicable

H. PURPOSE:

The X structure is used to contain common variables and information used to control the flow through the descriptor editor

TOPIC D.25 - MAINTENANCE

A. DATA SET NAME:

DEX which consists of the X external data structure minus all initialization values.

B. CREATED BY:

Not Applicable

C. TYPE OF FILE:

Table

D. ORGANIZATION:

FL/I Data Structure

E. KEY IDENTIFIER (CONTROL FIELD):

Not Applicable

F. RECORD LENGTH:

Not Applicable

G. BLOCKING FACTOR:

Not Applicable

H. PURPOSE:

The X structure is used to contain common variables and information used to control the flow through the descriptor editor.

I. PL/I DECLARATION:

THE X STRUCTURE IS A COLLECTION OF MINOR STRUCTURES AND SINGLE VARIABLES USED IN THE RUNNING OF THE DESCRIPTOR EDITOR. THESE MINOR STRUCTURES CONSIST OF PREDEFINED FLD, HDR, RECSEC SECURITY, AND SUPER STRUCTURES, AS WELL AS THE INPUT OUTPUT WORK AREAS FOR THE VARIOUS STRUCTURES. THE OTHER MINOR STRUCTURES ARE A LIST OF RESERVED FIELD NAMES AND A LIST OF FIELDS WHICH ARE TO BE DELETED FROM THE DESCRIPTOR FILE WHEN THE CURRENT DESCRIPTORS ARE FILED TO MAKE THE DESCRIPTOR FILE ACCURATE.

```
1 X          EXT CTL, /* EXTERNAL CONTROLLED      */
                /* BLOCKED VARIABLES.            */
```

THIS MINOR STRUCTURE IS THE PREDEFINED COMMENTS FIELD DESCRIPTOR.

```
3 FLD_COMMENTS LIKE FLD, /*COMMENTS FIELD DESCRIPTOR*/
```

THIS MINOR STRUCTURE IS THE PREDEFINED FREEFORM FIELD DESCRIPTOR.

```
3 FLD_FREEFORM LIKE FLD, /*USER ENTERED KEYWORDS    */
```

THIS MINOR STRUCTURE IS THE PREDEFINED RECORD SECURITY FIELD DESCRIPTOR.

```
3 FLD_RS      LIKE FLD,
                /* RECORD SECURITY DESCRIPTOR.*/
```

THIS MINOR STRUCTURE IS THE PREDEFINED BASE FOR A SUBFILE CONTROL FIELD DESCRIPTOR.

```
3 FLD_SUBCNTRL LIKE FLD,
```

THIS MINOR STRUCTURE IS THE PREDEFINED BASE FOR A SUBFILE KEY FIELD DESCRIPTOR.

```
3 FLD_SUBID    LIKE FLD,
```

THIS MINOR STRUCTURE IS THE PREDEFINED BASE FOR A SUBFILE PARENT KEY FIELD DESCRIPTOR.

```
3 FLD_SUBPK    LIKE FLD,
```

THIS MINOR STRUCTURE IS THE PREDEFINED HEADER DESCRIPTOR RECORD FOR THE ASSOCIATE FILE CONTAINING COMMENTS AND FREEFORM FIELD DESCRIPTORS.

```
3 HDR_ASSOC    LIKE HDR, /* HEADER FOR COMMENTS AND */
```

THIS MINOR STRUCTURE IS THE PREDEFINED HEADER DESCRIPTOR RECORD THE INDEX FILE ON WHICH THE FIELD FREEFORM IS INDEXED.

```
3 HDR_INDEX      LIKE HDR,
                  /* INDEX FILE HEADER FOR USER */
                  /* KEYWORDS STORED IN FREEFORM*/
```

THIS MINOR STRUCTURE IS A PREDEFINED INITIALIZED FLD STRUCTURE- IT IS USED TO INITIALIZE A NEWLY ALLOCATED FLD STRUCTURE.

```
3 INIT_FLD       LIKE FLD,
                  /* FIELD DESCRIPTOR INITIAL  */
                  /* VALUES.                  */
```

THIS MINOR STRUCTURE IS A PREDEFINED INITIALIZED HDR STRUCTURE. IT IS USED TO INITIALIZE A NEWLY ALLOCATED HDR STRUCTURE.

```
3 INIT_HDR       LIKE HDR,
                  /* HEADER DESCRIPTOR INITIAL */
                  /* VALUES.                  */
```

THIS MINOR STRUCTURE IS A PREDEFINED INITIALIZED SECURITY STRUCTURE. IT IS USED TO INITIALIZE A NEWLY ALLOCATED SECURITY STRUCTURE.

```
3 INIT_SECURITY, /* FIELD SECURITY STRUCTURE */
                  /* INITIAL VALUES.        */
5 #              BIN FIXED,
5 CODE (18) CHAR (8)
5 CHANGED (18) BIT (1),
5 FILLER CHAR (8), /* NEEDED FOR FLI BUG.    */
```

THIS MINOR STRUCTURE IS USED FOR ALL IO OPERATIONS TO AND FROM THE DESCRIPTOR FILE INVOLVING FIELD DESCRIPTOR RECORDS. ALL FIELD DESCRIPTOR INPUT FROM THE DESCRIPTOR FILE IS PLACED INTO THIS WORK AREA BEFORE BEING MOVED TO AN ALLOCATED FLD STRUCTURE. BEFORE OUTPUTTING TO A FIELD DESCRIPTOR ON THE DESCRIPTOR FILE, THE FIELD INFORMATION IS MOVED INTO THE IO_FLD STRUCTURE. THIS IS NECESSARY BECAUSE DBPAC REQUIRES ALL IO INTO AND FROM TO BE DONE FROM VARYING LENGTH CHARACTER STRINGS.

```
3 IO-FLD,        /* FIELD DESCRIPTOR WORK AREA */
                  /* STRUCTURE.                */
5 BACKWARD PTR,  /* BACKWARD FIELD POINTER.    */
5 FORWARD PTR,   /* FORWARD FIELD POINTER.     */
5 FLDNAME CHAR (8) VAR, /* FIELD NAME.                */
5 ASSOCFIL CHAR (1) VAR, /* ASSOCIATE FILE ID.         */
5 SUBFILE CHAR (1) VAR, /* SUBFILE IDENTIFIER.        */
```

```

5 INVFILE   CHAR (1) VAR,/* INVERTED FILE ID.      */
5 READONLY  CHAR (1) VAR,/* FIELD READ ONLY FLAG.  */
5 SUBCNTL   CHAR (1) VAR,/* FIELD IS A SUBFILE  */
                    /* CONTROL FIELD.      */
5 VARFLD    CHAR (1) VAR,/* VARGING LENGTH FIELD.*/
5 BITFLD    CHAR (1) VAR,/* FIXED LENGTH BIT     */
                    /* STRING OF LENGTH ONE.*/
5 NUMALIGN  CHAR (1) VAR,/* FIELD ALIGNMENT FLAG.*/
5 VARELT    CHAR (1) VAR,/* FIELD ELEMENTS OF    */
                    /* LENGTH FLAG.        */
5 UNIQUELT  CHAR (1) VAR,/* ELEMENTS UNIQUE FLAG.*/
5 INDEXEXT  CHAR (1) VAR,/* INDEX KEYS TO BE IN  */
                    /* INTERNAL OR EXTERNAL FORM */
                    /* FLAG.                */
5 FILEF     CHAR (1) VAR,/* BOUNDARY ALIGNMENT.  */
5 GENERCRT  CHAR (8) VAR,/* CCNVERSION RTN NAME. */
5 VALIDRTN  CHAR (8) VAR,/* VALIDATION RTN NAME. */
5 REFORMAT  CHAR (8) VAR,/* FORMATTING RTN NAME. */
5 SPARE     CHAR (8) VAR,/* UNUSED DESCRIPTOR FIELD.*/
5 FLDPOSIT  CHAR (2) VAR,/* FIELD POSITION VALUE.  */
5 FLDLEN    CHAR (2) VAR,/* FIELD LENGTH VALUE.   */
5 FLDLEN    CHAR (2) VAR,/* MAXIMUM FIELD LENGTH.*/
5 ELTLIM    CHAR (2) VAR,/* MAX NUMBER OF        */
                    /* ELEMENTS / FIELD.    */
5 DELTLIM   CHAR (2) VAR,/* MAXIMUM # OF ELEMENTS*/
5 ELTLEN    CHAR (2) VAR,/* ELEMENT LENGTH VALUE.*/
5 DELTLEN   CHAR (2) VAR,/* MAXIMUM ELEMENT LNTH*/
5 VALIDARG  PTR, /* POINTER TO VALIDATION */
                    /* ARGUMENT IF ANY.     */
5 NAMEFLD   PTR, /* POINTER TO LIST OF FIELD */
                    /* NAMES MAKING UP SUPER FIELD*/
5 SECURITY  PTR, /* POINTER TO FIFLD SECURITY */
                    /* CODES IF ANY.        */
5 BASEFLD   CHAR (8),/* SUBFIELD DEFINING BASE. */
5 OFFSET    BIN FIXED,/* OFFSET IN BASEFIELD    */
                    /* SUBFIELD IS TO START. */
5 FILE_LIST BIN FIXED,/* WHICH ENTRY IN FLD_TAB. */
5 FLDTYPE   BIN FIXED,/* TYPE OF FIELD.         */
5 CHANGED   (28) BIT (1),/* ONE FLAG FOR EACH ITEM */
                    /* IN FLD STRUCTURE. IF ON */
                    /* THEN PUT NEW VALUE IN  */
                    /* DESCRIPTOR FILE.      */
5 FILLER2   CHAR (8),/* NEEDED FOR PLI BUG.    */

```

THIS MINOR STRUCTURE IS USED FOR ALL IO OPERATIONS TO AND FROM THE DESCRIPTOR FILE INVOLVING HEADER RECORDS. ALL INPUT FROM THE DESCRIPTOR FILE INVOLVING HEADER RECORDS IS PLACED IN THE IO_HDR STRUCTURE BEFORE BEING MOVED TO AN ALLOCATED HDR STRUCTURE. TO OUTPUT A HEADER RECORD, THE INFORMATION IS MOVED INTO THE IO-HDR STRUCTURE BEFORE BEING PLACED ON THE FILE. THIS IS NECESSARY BECAUSE DBPAC REQUIRES ALL IO TO BE DONE IN INTO AND FROM VARYING LENGTH CHARACTER STRINGS.

```

3 IO_HDR, /* HEADER DESCRIPTOR WORK AREA */
5 BACKWARD PTR, /* BACKWARD HEADER POINTER. */
5 FORWARD PTR, /* FORWARD HEADER POINTER. */
5 SUFFIX CHAR (1) VAR, /* WHICH FILE THIS /*
/* HEADER BELONGS TO. */
5 FILETYPE CHAR (1) VAR, /* TYPE OF FILE INDICATR */
5 DESCRCT CHAR (2) VAR, /* NUMBER OF FIELD /*
/* DESCRIPTORS ON THIS FILE. */
5 BSELNGTH CHAR (2) VAR, /* TOTAL LENGTH OF FIXED */
/* FIELDS ON THIS FILE. */
5 DESCOK CHAR (1) VAR, /* DESCRIPTORS OK FLAG. */
5 SPANNED CHAR (1) VAR, /* THIS INDEX TO CONSIST */
/* OF SPANNED RECORDS FLAG. */
5 DATA CHAR (1) VAR, /* DATA IS ON FILE FLAG. */
5 MNTNABLE CHAR (1) VAR, /* FILE CAN BE /*
/* MAINTAINED FLAG. */
5 MNTNING CHAR (1) VAR, /* FILE BEING MAINTAINED */
/* FLAG. */
5 LOADABLE CHAR (1) VAR, /* FILE CAN BE LOADED. */
5 REMAINS CHAR (4) VAR, /* UNUSED DESCRIPTOR FLD */
5 RECSECFP CHAR (2) VAR, /* FILE HAS RECORD /*
/* SECURITY FLAG. */
5 RSECTYCD PTR, /* POINTER TO RECORD /*
/* SECURITY CODES IF ANY. */
5 CHANGED (13) BIT (1), /* ONE FLAG FOR EACH ITEM /*
/* IN HEADER STRUCTURE. IF /*
/* ON THEN PUT NEW VALUE /*
/* IN THE DESCRIPTOR FILE. */
5 FILLER CHAR (8), /* NEEDED FOR PLI BUG. */

```

THIS MINOR STRUCTURE IS USED FOR ALL IO OPERATIONS TO AND FROM THE DESCRIPTOR FILE INVOLVING FIELD SECURITY CODES.

```

3 IO SECURITY, /* FIELD SECURITY STRUCTURE. */
  5 # BIN FIXED, /* NUMBER OF SECURITY CODES */
    /* FOR THIS FIELD. */
  5 CODE (18) CHAR (8) VAR, /* USER PASSWORD. */
  5 CHANGED (18) BIT (1), /* ONE FLAG FOR EACH */
    /* SECURITY CODE. IF ON THEN */
    /* RESET THE NEW VALUE. */
  5 FILLER CHAR (8), /* NEEDED FOR PII BUG. */
3 GP, /* PARAMETERS TO GET FIELD */
    /* SUBROUTINE. */
  5 ALLOC_NEW BIT (1), /* ALLOCATE AND INITIALIZE A */
    /* NEW FLD STRUCTURE. */
  5 FLD_LEN BIN FIXED, /* MAXIMUM ALLOWABLE LENGTH */
    /* FOR THE FIELDNAME. */
  5 FLD_MSG CHAR (8), /* MSGID TO PROMPT FOR THE */
    /* FIELDNAME. */
  5 FLD# BIN FIXED, /* FOR AN EXISTING FIELD, */
    /* THE ENTRY IN FIELD STRUCTURE */
    /* ELSE 0. */

```

```

5 NEW_FLD      BIT (1), /* ON - GET A BRAND NEW FIELD*/
                  /* OFF - AN EXISTING FIELD. */
5 PRMPT_ERR BIT (1), /* VALUE TO SET TC.PROMPT.ERR*/
5 RESERVED BIT (1), /* A RETURNED VALUE INDICATING */
                  /* IF THE FIELDNAME IS */
                  /* RESERVED. */
5 RES_FLD      BIT (1), /* ON - RESERVED NAMES ARE */
                  /* ACCEPTABLE. */
                  /* OFF - RESERVED NAMES ARE */
                  /* NOT ACCEPTABLE. */
3 ADD_FLAG     BIT (1), /* IN ADD OR CHANGE COMMAND. */
3 ALPHA        CHAR (26), /* ALL ACCEPTABLE ALPHABETIC */
                  /* CHARACTERS. */
3 ALPHANUMERIC CHAR (36),
                  /* ALL ACCEPTABLE ALPHA- */
                  /* NUMERIC CHARACTERS. */
3 ARG_PTR      PTR, /* PTR TO VALIDATION ARGUMENT.*/
3 ASSOC_NAMES  CHAR (9), /* ALL POSSIBLE ASSOCIATE */
                  /* FILE ID'S. */
3 COMND_CALL   BIT (1), /* A COMMAND CALL OR AN */
                  /* INTERNAL CALL. */
3 COMND_NAME   CHAR (8), /* NAME OF COMMAND CALLED. */
3 ERR_FLAG     BIT (1), /* ERROR FLAG USED FOR */
                  /* INTER MODULE COMMUNICATION.*/
3 FIELDNAME    CHAR (8), /* FIELD TO BE PROCESSED. */
3 FIELDTYPE (0:10) CHAR (2), /* ALL VALID FIELD TYPES.*/
3 FIELD_PTR    PTR, /* PTR TO FIELD NAME STRUCTURE*/
3 FLD_IASST (60) PTR, /* PTRS TO LAST FIELD ENTRY */
                  /* IN EACH FIELD STRING. */
3 FLD_PTR      PTR, /* PTR TO FIELD DESCRIPTOR. */
3 FLD_TAB (60) PTR, /* PTRS TO FIRST FIELD ENTRY */
                  /* IN EACH FIELD STRING. */
3 FLDTYPE      BIN FIXED, /* FIELD TYPE USED BY ADD. */
3 FSEC_PTR     PTR, /* PTR TO FIELD SECURITY */
                  /* STRUCTURE. */
3 HDR_PTR      PTR, /* PTR TO FILE DESCRIPTOR. */
3 HEAD_TAB (36) PTR, /* ONE PTR FOR EACH HEADER. */
3 HEX_CHARS    CHAR (16),
                  /* ALL ACCEPTABLE HEXADECEMAL */
                  /* CHARACTERS. */
3 INDEX_NAMES  CHAR (16),
                  /* LIST OF ALL POSSIBLE INDEX */
                  /* FILE ID'S. */
3 IOAREA       CHAR (256) VAR, /* COMMON TERMINAL INPUT */
                  /* OUTPUT AREA. */
3 LOAD_FILE    CHAR (1), /* ID OF FILE TO LOAD FROM. */
3 LOAD_ONE     BIT (1), /* LOAD JUST ONE RECORD. */
3 LVA         BIN FIXED, /* LENGTH OF VALIDATION */
                  /* ARGUMENT. */
3 PAT_FILE     CHAR (1), /* ID OF FILE BEING WORKED */
                  /* ON BY REVIEW - PATCH. */
3 PAT_FIELD    CHAR (8), /* NAME OF FIELD BEING */
                  /* WORKED ON BY REVIEW-PATCH. */

```

```

3 REV_MODE      BIT (1), /* IN REVIEW OR UPDATE MODE. */
3 RSEC_PTR      PTR,    /* PTR TO RECORD SECURITY    */
                      /*      CODES.                */
3 SAVE_STRING   CHAR (150) VAR, /* AREA TO BUILD      */
                      /*      COMMAND STRINGS.    */
3 SUBFILE_NAMES CHAR (10),
                      /* LIST OF ALL POSSIBLE  */
                      /*      SUP-FILE ID'S.    */
3 SUFFIX        CHAR (36),
                      /* ALL POSSIBLE FILE     */
                      /*      IDENTIFIER SUFFICES. */
3 SUPER_PTR     PTR,    /* PTR TO SUPERFIELD COMPONENT */
3 TRANS,        /* TRANSITORY CALL LABELS. */
    5 CALL      CHAR (8), /* ROUTINE TO BE CALLED      */
    5 RET       CHAR (8), /* ROUTINE TO RETURN TO.    */

```

THIS SUBSEQUENT PART OF THE X STRUCTURE IS SEPARATED FROM THE REST OF THE X ITEMS AS THIS PART OF X IS THE PART THAT MUST BE SAVED WHEN USING THE CHKPOINT COMMAND. THIS PREVIOUS INFORMATION OF X NEED NOT BE SAVED, AS IT IS SETUP PROPERLY WHENEVER X IS ALLOCATED, OR THOSE ITEMS WHOSE VALUES MATTERS NOT BETWEEN COMMAND EXECUTION.

```

3 CHKPOINT_RECLN BIN (31) FIXED,
                      /* OUTPUT RECORD LENGTH FOR */
                      /*      ASMPUT ROUTINE. IT IS SET */
                      /*      SET TO THE LENGTH OF THE */
                      /*      X STRUCTURE THAT MUST BE */
                      /*      SAVED WHEN CHECKPOINT IS */
                      /*      EXECUTED.                */

```

THIS MINOR STRUCTURE IS THE PREDFFINED RECLN FIELD DESCRIPTOR. IT IS PLACED IN IN THIS PART OF X BECAUSE IT MAY HAVE FIELD SECURITY APPLIED TO IT.

```

3 FLD_RECLN     LIKE FLD,

```

THIS MONOR STRUCTURE IS A LIST OF RESERVED FIELDNAMES. THE USER MAY NOT DEFINE BY USE OF THE ADD, SUPERFLD, CREATESUB, ADDLIKE, AND RENAME CCMMANDS A FIELD DESCRIPTOR WITH A FIELDNAME THAT APPEARS IN THIS TABLE.

```

3 RESERVED,        /* LIST OF RESERVED FIELDNAMES */
    5 LAST_#       BIN FIXED INIT (14),
                      /* INDEX OF LAST ENTRY.      */
    5 FIELDNAME (40) CHAR (8), /* RESERVED FIELDNAMES */
3 ASSOC_LIST       CHAR (9), /* LIST OF ASSOCIATE FILE */
                      /*      ID'S AVAILABLE FOR */
                      /*      ASSIGNMENT.        */
3 CREATEF         BIT (1), /* CREATE-UPDATE MODE FLAG. */
3 DATAPLEX        CHAR (6), /* FILE BEING DEFINED.      */

```

```

3 DELETE_FILES  CHAR (36), /* LIST OF DESCRIPTOR      */
                      /* REGIONS TO BE DELETED FROM */
                      /* DISC.                      */
3 EXIST_FILES   CHAR (36), /* FILE IDS OF ALL FILES */
                      /* EXISTING ON DISC.          */
3 FILE_EXISTS   BIT (1), /* DESCRIPTOR FILE EXISTS. */
3 INDEX_LIST    CHAR (16), /* LIST OF UNASSIGNED INDEX*/
                      /* FILE ID'S.                */
3 #FN           BIN FIXED, /* NUMBER OF ENTRIES IN   */
                      /* FIELD STRUCTURE.       */
3 LOAD_ERROR    BIT (1), /* ERROR IN LOADING DESCPTR*/
3 NEED_FILE     BIT (1), /* USER SHOULD FILE TO SAVE. */
3 SUBFILE_LIST  CHAR (10), /* LIST OF ALL UNASSIGNED  */
                      /* SUB_FILE ID'S.         */

```

THIS MINOR STRUCTURE IS USED TO STORE THAT INFORMATION NECESSARY TO THE EXECUTION OF MODULES THAT CAN HAVE PAGEABLE INFORMATION DISPLAYS.

```

3 PAGE_INFO,
  5 RTN          LABEL, /* WHAT ROUTINE TO PAGE.    */
  5 RTN_NAME     CHAR (8),
                      /* THE ROUTINE NAME THE PAGING*/
                      /* MODULE IS TO CALL.        */
  5 PTR          PTR, /* ADDRESS OF STRUCTURE BEING */
                      /* DISPLAYED.                */
  5 DIR          CHAR (1), /* PAGING DIRECTION.        */
  5 FILE_ID      CHAR (1), /* SUFFIX OF FILE BEING      */
                      /* REVIEWED.                 */
  5 FLD_NAME     CHAR (8),
                      /* NAME OF FIELD BEING REVIEWED */
  5 !OTE!        BIN FIXED, /* LAST ITEM PUT ON SCREEN-*/
  5 ITEMS        BIN FIXED, /* # OF ITEMS TO DISPLAY.    */
  5 LIMIT        BIN FIXED, /* ALL ITEMS AFTER LIMIT TO*/
                      /* BE DISPLAYED ONE PER LINE. */
  5 #            BIN FIXED, /* # OF PAGE BEING DISPLAYED*/
  5 LAST         BIN FIXED, /* LAST ENTRY USED.         */
  5 START (100) BIN FIXED, /* ITEM # USED TO START     */
                      /* EACH PAGE.               */

```

THIS MINOR STRUCTURE IS USED TO STORE THOSE FIELD NAMES AND IDS OF THE DESCRIPTOR REGIONS IN WHICH THEY APPEAR THAT MUST BE DELETED FROM THE DESCRIPTOR FILE THE NEXT TIME A FILE IS DONE.

```

3 DELETE,          /* LIST OF FIELD NAMES TO BE */
                      /* DELETED FROM THE DISC.    */
  5 KEY_NAME       CHAR (8), /* ANCHOR KEY NAME IF        */
                      /* ANCHOR KEY NAME HAS BEEN  */
                      /* CHANGED.                  */
  5 #              BIN FIXED, /* NUMBER OF FIELDS          */

```

```

/*      TO BE DELETED.      */
5 A (100),
  7 FIELD CHAR (8),/* NAMES OF FIELDS TO BE */
    /*      DELETED.      */
  7 IDS   CHAR (4);/* IDS OF FILES ON WHICH */
    /*      THE FIELD APPEARS.      */
```

TOEIC D.26 - MAINTENANCE

A. DATA SET NAME:

DEHDR which consists of the structures HDR and HDR_STRING

B. CREATED BY:

Not Applicable

C. TYPE OF FILE:

Table

D. ORGANIZATION:

PL/I Data Structure

E. KEY IDENTIFIER (CONTROL FIELD):

Not Applicable

F. RECORD LENGTH:

Not Applicable

G. BLOCKING FACTOR:

Not Applicable

H. PURPOSE:

The HDR structure is used to contain the information describing a file descriptor. HDR_STRING is a character string used to overlay the HDR structure.

I. PL/I DECLARATION:

THE HDR STRUCTURE IS USED STORE THE INFORMATION
DEFINING A FILE DESCRIPTOR.

```

1 HDR      BASED (X.HDR_PTR), /* FILE DESCRIPTOR      */
           /* STRUCTURE.                */
3 BACKWARD PTR,      /* BACKWARD HDR POINTER.    */
3 FORWARD  PTR,      /* FORWARD HDR POINTER.     */
3 SUFFIX   CHAR (1), /* WHICH FILE THIS HEADER   */
           /* BELONGS TO.              */
3 FILETYPE CHAR (1)  /* TYPE OF FILE INDICATOR.  */
3 DESCRCT  BIN FIXED, /* NUMBER OF FIELD DESCRIPTORS*/
           /* ON THIS FILE.            */
3 BSELNGTH BIN FIXED, /* TOTAL LENGTH OF FIXED    */
           /* FIELDS ON THIS FILE.     */
3 DESCOK   CHAR (1), /* DESCRIPTORS CK FLAG.     */
3 SPANNED  CHAR (1), /* THIS INDEX TO CONSIST OF */
           /* SPANNED RECORDS.        */
3 DATA    CHAR (1), /* DATA ON FILE SWITCH.    */
3 MNTNABLE CHAR (1), /* FILE CAN BE MAINTAINED FLAG*/
3 MNTNING  CHAR (1), /* FILE BEING MAINTAINED FLAG*/
3 LOADABLE CHAR (1), /* WHETHER OR NOT TO PLACE  */
           /* DATA ON THIS FILE.     */
3 REMAINS  CHAR (8), /* UNUSED HDR DESCRIPTOR FIELD*/
3 RECSECFP BIN FIXED, /* FILE HAS RECORD SECURITY. */
3 RSECTYCD PTR,      /* POINTER TO RECORD SECURITY */
           /* CODES IF ANY.           */
3 CHANGED (13) BIT (1), /* ONE FLAG FOR EACH ITEM IN */
           /* HEADER STRUCTURE. IF ON   */
           /* THEN PUT NEW VALUE IN    */
           /* DESCRIPTOR FILE.        */
3 FILLER   CHAR (8); /* NEEDED FOR PLI BUG.      */

```

THIS STRUCTURE IS A CHARACTER STRING OVERLAY ON THE HDR
STRUCTURE. IT IS USED FOR MAKING COPIES OF THE HDR
STRUCTURE.

```

DCL HDR_STRING CHAR (46), BASED (X.HDR_PTR);
           /* HDR STRUCTURE OVERLAY.      */

```

TOPIC D.27 - MAINTENANCE

A. DATA SET NAME:

CARDIN Freeform Parameter File

B. CREATED BY:

User

C. TYPE OF FILE:

SYSIN Input

D. ORGANIZATION:

Sequential (QSAM)

E. KEY IDENTIFIER (CONTROL FIELD):

Not Applicable

G. FLOCKING FACTOR:

Not Applicable

H. PURPOSE:

This input card file contains the parameters needed for each individual maintenance job. Parameters are entered on cards in a freeform format. Parameters must be Keyworded and separated by a comma. Element parameters must be enclosed in left and right parens. No embedded blanks are allowed but Keyword can start in any position in the card. Values must not be split on two different cards but element parameters may be continued on next card if preceded with a Keyword. Following is an example of parameter cards:

```
//CARDIN      DD      *
      FILENAME=DE2TDB
      FIELD=(EMPNAME,EMPAGE,EMPSEX)
      ANCHOR=YES,ASSOCIAT=NO,INDEX=NC
```

TOPIC D.28 - MAINTENANCE

A. DATA SET NAME:

PRTOUT PRINT FILE

B. CREATED BY:

Maintenance Modules

C. TYPE OF FILE:

Printer Output

D. ORGANIZATION:

Sequential (QSAM)

E. KEY IDENTIFIER (CONTROL FIELD):

Not Applicable

F. RECORD LENGTH:

Fixed 133 bytes (first byte print control character).

G. BLOCKING FACTOR:

Not Applicable

H. PURPOSE:

This is a printer output file for maintenance modules to display diagnostic information or program status information such as record counts. DD card should be:

```
//PRTOUT DD SYSOUT=A,DCB=(RECFM=FA,IRECL=133,
// BLKSIZE=133)
```

TOPIC E.1 - TERMINAL SUPPORT

A. DATA SET NAME:

TSPL/I Diagnostics

B. CREATED BY:

TS Preprocessor Function

C. TYPE OF FILE:

(4) Table

D. ORGANIZATION:

Keyed List

E. KEY IDENTIFIER (CONTROL FIELD):

Each diagnostic comment has an identification key having the form: '---ERROR---nn' where nn is a unique identification number.

F. RECORD LENGTH:

Variable

G. BLOCKING FACTOR:

Not Applicable

H. PURPOSE:

TSPL/I Diagnostic comments are generated into mainline source programs by the TS preprocessor function (see Section IV, Topic E.1 of the DWE). They are tabulated here with additional notes for reference. In Paragraph I, single quotes denote that characters from the TS preprocessor function or its argument are filled into the message to make its meaning clearer.

I. TSPL/I DIAGNOSTIC COMMENTS:

TS001 MISSING ARGUMENT ON TSPL/I REFERENCE.

Severe error - a TS preprocessor function reference has no parenthesized argument.

TS002 TSPL/I ARGUMENT DOES NOT BEGIN WITH A ' ('.

Severe error - a TS preprocessor function reference

does not begin with double left parentheses. Processing of this TS reference was abandoned because the closing right parenthesis would not be able to be found.

TSC03 MISSING DELIMITER IN TSPL/I STATEMENT.

Severe error - the right parenthesis at the end of a TS preprocessor function reference has been encountered unexpectedly.

TSC04 STATEMENT HAS A MISSING ';'.

Severe error - the right parenthesis at the end of a TS preprocessor function reference has been encountered unexpectedly.

TSC05 STATEMENT FOUND FOLLOWING FINISH.

Severe error - the statement has been ignored because it follows the TS ((FINISH;)) reference.

TSC06 STATEMENT CONTAINS EXCESS '('(s).

Severe error - the statement semicolon has been found, but the parentheses are unbalanced. The statement was ignored.

TSC07 STATEMENT KEYWORD UNKNOWN.

Severe error - an unknown word was found as the first word of a TSPL/I statement. The statement was ignored.

TSC08 'text' STATEMENT CONTAINS INVALID SYNTAX.

The statement type identified by 'text' was found to contain invalid syntax. The statement was ignored.

TSC09 EXTRANEIOUS TEXT IGNORED.

This message merely means that part of the statement was ignored.

TSC10 IMPROPER OR MULTIPLE ENABLE STATEMENTS.

An improper placement of or multiple use of an enable statement has been encountered. The statement was ignored.

** '---NNNNN---** : TSPL/I ERRORS.'

The finish statement has been processed and NNNNN
errors were previously detected.

TOEIC E.2 - TERMINAL SUPPORT

A. DATA SET NAME:

Terminal Control Block

B. CREATED BY:

TS Preprocessor Function

C. TYPE OF FILE:

(4) Table

D. ORGANIZATION:

Linear Structure of Fields

E. KEY IDENTIFIER (CONTROL FIELD):

TC

The terminal control block is an automatic internal data table.

F. RECORD LENGTH:

236 Bytes (Hexidecimal EC)

This is the length of the whole control block including the dope vectors.

G. BLOCKING FACTOR:

Not Applicable

H. PURPOSE:

The TC control block is used for communication between mainline programs and TSSUP. The declaration is generated by the TS preprocessor function. For TSPL/I statements in the mainline, the TS preprocessor function generates statements that post fields in TC, such as a prompt message key. At execution time, TSSUP refers to fields in TC and posts error code fields in TC which may subsequently be referenced in the mainline.

I. PL/I DECLARATION:

```

/*      TERMINAL CONTROL BLOCK (TC) FOR (TS2) TSPL/I      */
DECLARE
1 TC,                                     /*DEFINE THE TC STRUCTURE */
  2 FUNCTION CHAR(8),                     /*TS FUNCTION IDENTIFIER  */
                                          /*SET BY TS PREPROCESSOR  */
                                          /*ENTRY '=ENTRY          */
                                          /*READ  '=READ           */
                                          /*WRITE '=WRITE          */
                                          /*FLUSH '=FLUSH          */
                                          /*PUT   '=PUT            */
                                          /*PROMPT-C'=COMMAND PROMPT*/
                                          /*PROMPT-D'=DATA PROMPT  */
                                          /*PROMPT-M'=MESSAGE      */
2 PAGING_ENTRY CHAR(8),                  /*TS PAGING ENTRY POINT   */
                                          /*SET BY TS PREPROCESSOR  */
                                          /*TO NAME OF THE CURRENT  */
                                          /*MODULE'S PAGING ENTRY   */
2 LINE_SIZE FIXED BIN(15),               /*TS LINE WIDTH (BYTES)   */
                                          /*SET BY TSSUP ON ENTRY   */
2 INPUT,                                  /*TS SCREEN INPUT FIELDS  */
  3 ERROR BIT(1),                         /*READ ERROR BIT SWITCH   */
                                          /*SET BY TSSUP AFTER READ */
                                          /*'0'=NO ERROR '1'=ERROR */
  3 EXTRA_BITS BIT(7),                   /*RESERVED BIT SWITCHES   */
2 OUTPUT,                                 /*TS SCREEN OUTPUT FIELDS */
  3 SIZE FIXED BIN(15),                   /*OUTPUT AREA SIZE (LINES)*/
                                          /*SET BY TSSUP ON ENTRY   */
  3 INDENT FIXED BIN(15),                  /*INDENTATION COLUMN NUMBER*/
                                          /*SET BY USER AT ANYTIME  */
  3 WRITTEN FIXED BIN(15),                 /*PUT OUTPUT COUNT (BYTES)*/
                                          /*SET BY TSSUP ON OVERFLOW*/
                                          /*IF AUTO_WRITE IS SET ON */
  3 DIRECTION BIT(1),                     /*PUT DIRECTION BIT SWITCH*/
                                          /*SET BY TS PREPROCESSOR  */
                                          /*'0'=FORWARD '1'=BACKWARD*/
  3 PUT_PARTIAL BIT(1),                   /*PUT OUTPUT MODE SWITCH  */
                                          /*SET BY USER AT ANYTIME  */
                                          /*'0'=PUT FULL RECORD ONLY*/
                                          /*'1'=PUT PARTIAL RECORD  */
  3 AUTO_WRITE BIT(1),                     /*PUT END OF BUFFER SWITCH*/
                                          /*SET BY USER AT ANYTIME  */
                                          /*'0'=RETURN TO USER      */
                                          /*'1'=AUTOMATIC WRITE     */
  3 WORD_BREAK BIT(1),                     /*PUT LINE SPLIT SWITCH   */
                                          /*SET BY USER AT ANYTIME  */
                                          /*'0'=TRUNCATE AT LINE END*/
                                          /*'1'=BREAK AT LAST WORD  */
  3 OVERFLOW BIT(1),                       /*PUT OVERFLOW BIT SWITCH */

```

```

3 CONTINUATION BIT(1), /*SET BY TSSUP WHEN PUT */
/*CAUSES BUFFER OVERFLOW */
/*'0'=NO OVERFLOW */
/*'1'=OVERFLOW */
3 POSITION BIT(1), /*PUT CONTINUATION SWITCH */
/*SET BY USER WHEN HE IS */
/*PUTTING CONTINUED DATA */
/*'0'=NO CONTINUATION */
/*'1'=PUT CONTINUED DATA */
3 MORE_DATA BIT(1), /*PUT POSITIONING SWITCH */
/*SET BY TS PREPROCESSOR */
/*'0'=PUT TO NEXT LINE */
/*'1'=PUT TO TOP OF SCREEN */
/*SCREEN OVERFLOW SWITCH */
/*SET BY THE USER WHEN HE */
/*HAS MORE DATA TO OUTPUT */
/*VIA THE PAGING MECHANISM */
/*'0'=NO MORE DATA REMAINS */
/*'1'=MORE DATA AVAILABLE */
2 PROMPT, /*TS SCREEN PROMPT FIELDS */
3 SIZE FIXED BIN(15), /*PROMPT AREA SIZE (LINES) */
/*SET BY TSSUP ON ENTRY */
3 MESSAGE_KEY CHAR(8), /*KEY OF CURRENT MESSAGE */
/*SET BY TS PREPROCESSOR */
3 KEYWORD CHAR(8), /*KEYWORD FOR DATA PROMPT */
/*SET BY TS PREPROCESSOR */
3 BYPASS BIT(1), /*PROMPTING BYPASS SWITCH */
/*SET BY USER AT ANYTIME */
/*'0'=PROMPT IF NO DATA */
/*'1'=RETURN NULL VALUE */
3 ERROR BIT(1), /*PROMPTING ERROR SWITCHZZZ*/
/*SET BY USER WHEN A DATA */
/*ERROR FORCES REPROMPTING */
/*'0'=PROCESS NORMALLY */
/*'1'=REPROMPT FOR DATA */
3 TRUNCATION BIT(1), /*DATA TRUNCATION SWITCH */
/*SET BY TSSUP FOR PROMPT */
/*'0'=NO TRUNCATION */
/*'1'=DATA TRUNCATED */
3 DEFAULT BIT(1), /*DEFAULT VALUE BIT SWITCH */
/*SET BY TSSUP FOR PROMPT */
/*'0'=DATA ENTERED BY USER */
/*'1'=DATA WAS A DEFAULT */
3 UNQUOTED BIT(1), /*UNQUOTED UNQUOTED BIT SWITCH */
/*SET BY TSSUP FOR PROMPT */
/*'0'=NORMAL DATA VALUE */
/*'1'=QUOTED STRING */
3 MORE_DATA BIT(1), /*PARENTHIZED LIST SWITCH */
/*SET BY TSSUP FOR PROMPT */
/*'0'=LAST DATA VALUE */
/*'1'=MORE VALUES FOLLOW */
3 SKIP BIT(1), /*SKIP INPUT PARSING BIT */
/*SET BY THE USER WHEN HE */

```

3 PAGE BIT(1);

```

/*WISHES TO BYPASS PARSING */
/*'0'=DC NORMAL PARSING    */
/*'1'=SKIP NORMAL PARSING  */
/*PAGING CONTROL SWITCH    */
/*'0'=IGNORE PAGING ENTRY  */
/*'1'=ALTER PAGING ENTRY   */

```

TCBIC E.3 - TERMINAL SUPPORT

A. DATA SET NAME:

TSTEXT - Terminal Control Block Declaration

B. CREATED BY:

Included by TS processor function.

C. TYPE OF FILE:

Table

D. ORGANIZATION:

PL/I Source Statements

E. KEY IDENTIFIER (CONTROL FIELD):

Not Applicable

F. RECORD LENGTH:

92 bytes

G. BLOCKING FACTOR:

Not Applicable

H. PURPOSE:

The purpose of the TSTEXT is to define the terminal control block (TC) within every program using it. This enables the programmer and the preprocessor to refer to the fields of the TC block in order to specify the various functions and options needed by the program.

I. PL/I DECLARATION:

DECLARE

```

    TSFLUSH ENTRY(),           /* FLUSH ENTRY POINT          */
    TSREAD ENTRY(,CHAR(*) VAR),/* READ ENTRY POINT          */
    TSWRITE ENTRY(,CHAR(*) VAR),/* WRITE ENTRY POINT         */
    TSPUT ENTRY(,CHAR(*) VAR, /* PUT ENTRY POINT           */
                CHAR(*) VAR),/* DEFINITION                 */
    ( TSPRMT,                  /* COMMAND PROMPT ENTRY      */
      TSPRMTD,                 /* DATA PROMPT ENTRY        */
      TSPRMTM ) ENTRY(,CHAR(*) VAR, /* MESSAGE ENTRY POINT      */
    CHAR(*) VAR, CHAR(*) VAR, CHAR(*) VAR, CHAR(*) VAR,
    CHAR(*) VAR, CHAR(*) VAR, CHAR(*) VAR, CHAR(*) VAR,
    CHAR(*) VAR, CHAR(*) VAR, CHAR(*) VAR, CHAR(*) VAR,
```

```

CHAR(*) VAR, CHAR(*) VAR, CHAR(*) VAR, CHAR(*) VAR);
DECLARE
    1 TC,
2 FUNCTION CHAR(8),
    2 PAGING_ENTRY CHAR(8),
    2 LINE_SIZE FIXED BIN(15),
    2 INPUT,
        3 ERROR BIT(1),
        3 EXTRA_BITS BIT(7),
    2 OUTPUT,
        3 SIZE FIXED BIN(15),
        3 INDENT FIXED BIN(15),
        3 WRITTEN FIXED BIN(15),
        3 DIRECTION BIT(1),
        3 PUT_PARTIAL BIT(1),
        3 AUTO_WRITE BIT(1),
        3 WORD_BREAK BIT(1),
        3 OVERFLOW BIT(1),
        /* DEFINE THE TC STRUCTURE */
        /* TS FUNCTION IDENTIFIER */
        /* SET BY TS PREPROCESSOR */
        /* 'ENTRY' =ENTRY */
        /* 'READ' =READ */
        /* 'WRITE' =WRITE */
        /* 'FLUSH' =FLUSH */
        /* 'PUT' =PUT */
        /* 'PROMPT-C'=COMMAND PROMPT */
        /* 'PROMPT-D'=DATA PROMPT */
        /* 'PROMPT-M'=MESSAGE */
        /* TS PAGING ENTRY POINT */
        /* SET BY TS PREPROCESSOR */
        /* TO NAME OF THE CURRENT */
        /* MODULE'S PAGING ENTRY */
        /* TS LINE WIDTH (BYTES) */
        /* SET BY TSSUE CN ENTRY */
        /* TS SCREEN INPUT FIELDS */
        /* READ ERROR BIT SWITCH */
        /* SET BY TSSUE AFTER READ */
        /* '0'=NC ERROR '1'=ERROR */
        /* RESERVED BIT SWITCHES */
        /* TS SCREEN OUTPUT FIELDS */
        /* OUTPUT AREA SIZE (LINES) */
        /* SET BY TSSUE ON ENTRY */
        /* INDENTATION COLUMN NUMBER */
        /* SET BY USER AT ANYTIME */
        /* PUT OUTPUT COUNT (BYTES) */
        /* SET BY TSSUE ON OVERFLOW */
        /* IF AUTO_WRITE IS SET ON */
        /* PUT DIRECTION BIT SWITCH */
        /* SET BY TS PREPROCESSOR */
        /* '0'=FORWARD '1'=BACKWARD */
        /* PUT OUTPUT MODE SWITCH */
        /* SET BY USER AT ANYTIME */
        /* '0'=PUT FULL RECORD ONLY */
        /* '1'=PUT PARTIAL RECORD */
        /* PUT END OF BUFFER SWITCH */
        /* SET BY USER AT ANYTIME */
        /* '0'=RETURN TO USER */
        /* '1'=AUTOMATIC WRITE */
        /* PUT LINE SPLIT SWITCH */
        /* SET BY USER AT ANYTIME */
        /* '0'=TRUNCATE AT LINE END */
        /* '1'=BREAK AT LAST WORD */
        /* PUT OVERFLOW BIT SWITCH */
        /* SET BY TSSUE WHEN PUT */
        /* CAUSES BUFFER OVERFLOW */
        /* '0'=NC OVERFLOW */
        /* '1'=OVERFLOW */

```

```

3 CONTINUATION BIT(1), /* PUT CONTINUATION SWITCH */
/* SET BY USER WHEN HE IS */
/* PUTTING CONTINUED DATA */
/* '0'=NC CONTINUATION */
/* '1'=PUT CONTINUED DATA */
3 POSITION BIT(1), /* PUT POSITIONING SWITCH */
/* SET BY TS PREPROCESSOR */
/* '0'=PUT TO NEXT LINE */
/* '1'=PUT TO TOP OF SCREEN */
3 MORE_DATA BIT(1), /* SCREEN OVERFLOW SWITCH */
/* SET BY THE USER WHEN HE */
/* HAS MORE DATA TO OUTPUT */
/* VIA THE PAGING MECHANISM */
/* '0'=NC MORE DATA REMAINS */
/* '1'=MORE DATA AVAILABLE */
2 PROMPT, /* TS SCREEN PROMPT FIELDS */
3 SIZE FIXED BIN(15), /* PROMPT AREA SIZE (LINES) */
/* SET BY TSSUP ON ENTRY */
3 MESSAGE_KEY CHAR(8), /* KEY OF CURRENT MESSAGE */
/* SET BY TS PREPROCESSOR */
3 KEYWORD CHAR(8), /* KEYWORD FOR DATA PROMPT */
/* SET BY TS PREPROCESSOR */
3 BYPASS BIT(1), /* PROMPTING BYPASS SWITCH */
/* SET BY USER AT ANYTIME */
/* '0'=PROMPT IF NO DATA */
/* '1'=RETURN NULL VALUE */
3 ERROR BIT(1), /* PROMPTING ERROR SWITCH */
/* SET BY USER WHEN A DATA */
/* ERROR FORCES REPROMPTING */
/* '0'=PROCESS NORMALLY */
/* '1'=REPROMPT FOR DATA */
3 TRUNCATION BIT(1), /* DATA TRUNCATION SWITCH */
/* SET BY TSSUP FOR PROMPT */
/* '0'=NO TRUNCATION */
/* '1'=DATA TRUNCATED */
3 DEFAULT BIT(1), /* DEFAULT VALUE BIT SWITCH */
/* SET BY TSSUP FOR PROMPT */
/* '0'=DATA ENTERED BY USER */
/* '1'=DATA WAS A DEFAULT */
3 QUOTED BIT(1), /* QUOTED QUOTED BIT SWITCH */
/* SET BY TSSUP FOR PROMPT */
/* '0'=NORMAL DATA VALUE */
/* '1'=QUOTED STRING */
3 MORE_DATA BIT(1), /* PARENTHIZED LIST SWITCH */
/* SET BY TSSUP FOR PROMPT */
/* '0'=LAST DATA VALUE */
/* '1'=MORE VALUES FOLLOW */
3 SKIP BIT(1), /* SKIP INPUT PARSING BIT */
/* SET BY THE USER WHEN HE */
/* WISHES TO BYPASS PARSING */
/* '0'=DC NORMAL PARSING */
/* '1'=SKIP NORMAL PARSING */
3 PAGE BIT(1); /* PAGING CONTROL SWITCH */

```

```
/* 'C'=IGNORE PAGING ENTRY */  
/* '1'=ALTER PAGING ENTRY  */
```

TOPIC F.1 - DATA RETRIEVAL

A. DATA SET NAME

RETDATA - Retrieval Data Table

B. CREATED BY:

DBINIT

C. TYPE OF FILE:

Table

D. ORGANIZATION

PL/I Data Structure

E. KEY IDENTIFIER (CONTROL FIELD):

Not Applicable

F. RECORD LENGTH:

Not Applicable

G. BLOCKING FACTOR:

Not Applicable

H. PURPOSE

1. RETDATA EXTERNAL CONTROLLED.
This table contains data fields unique to the retrieval sub-system and referenced by various modules of that sub-system.
2. NAME CHAR(50) VARYING.
This field contains the name of the user.
2. ADDRESS CHAR(100) VARYING.
This field contains the address of the user.

I. PL/I DECLARATION

```
/*          NASIS SYSTEM RETRIEVAL DATA TABLE          */
```

DCL

```
1 RETDATA EXTERNAL CONTROLLED, /*DEFINE RETRIEVAL DATA */
2 NAME CHAR(50) VAR,           /* USER'S NAME      */
2 ADDRESS CHAR(100) VAR;       /* USER'S ADDRESS  */
```

TOFIC F.2 - DATA RETRIEVAL

A. DATA SET NAME:

EXPAND Display Format

B. CREATED BY:

EXPAND (DBXPNE)

C. TYPE OF FILE:

(3) Terminal Communication

D. ORGANIZATION:

Character Display Screen

E. KEY IDENTIFIER (CONTROL FIELD):

Not Applicable

F. RECORD LENGTH:

Variable (Enter output area of the screen or pseudo screen)

G. BLOCKING FACTOR:

Not Applicable

H. PURPOSE:

This is a series of on-line output displays produced by the EXPAND ccommand giving the user full access to the inverted indexes of a data base assisting him in an on-line search for information.

The display is adapted to the size of the display screen being used. If the end of the inverted index or the end of the range of E-numbers (000-999) is encountered in either direction, a line such as

```
{--TOP OF PAGING SEQUENCE--}  
{--END OF INDEX----
```

is displayed in the appropriate row. The primary term is always regenerated on the appropriate row when multiple paging operations are done in either direction even if the primary term is not found in the inverted index.

SAMPLE EXPAND DISPLAY

```
SYSTEM: -ENTER:
USER:  expand asm,language
SYSTEM:  LINE XREFS  LANGUAGE(S)
        -E100      28  ASM
          E101       6  ENG
          E102      12  N/A
          E103      43  PLI
          E104       4  TSS
        ****          (---END OF INDEX---)
```

TOPIC F.3 - DATA RETRIEVAL

A. DATA SET NAME:

SELECT Display Format

B. CREATED BY:

SELECT (DBSLCT and DBSETU)

C. TYPE OF FILE:

(3) Terminal communication

D. ORGANIZATION:

Character Display Screen

E. KEY IDENTIFIER (CONTROL FIELD):

Not Applicable

F. RECORD LENGTH:

480 Byte typical - 40 column, 22 line output area apart from the prompting area.

G. BLOCKING FACTOR:

Not Applicable

H. PURPOSE:

This is the output generated by the SELECT command. DBSLCT calls the DBPSET entry point of DBSETU to post the users new set and DBSETU sends this display to the prompt area of the screen.

SELECT COMMAND SCREEN DISPLAY

aa bbbbt ccccccc

or

aa bbbbb (FROM: dddddd) ccccccc

where:

aa	= set number
bbbbb	= number of references
ccccccc	= SELECT expression
dddddd	= control field name, if applicable

TOHIC F.4 - DATA RETRIEVAL

A. DATA SET NAME:

DISPLAY Display Format

B. CREATED BY:

DISPLAY (DBDSPL)

C. TYPE OF FILE:

(3) Terminal Communication

D. ORGANIZATION:

Character Display Screen

E. KEY IDENTIFIER (CONTROL FIELD):

Not Applicable

F. RECORD LENGTH:

480 Bytes typical - 40 column, 12 line output area
apart from the prompting area.

G. BLOCKING FACTOR:

Not Applicable

H. PURPOSE:

This is a series of on-line output displays produced by the DISPLAY command giving the user full access to the anchor and associated files of a data base assisting him in an on-line search for information. Each screen image is built in a PAGTAB buffer and then transmitted in a single output operation to the display screen. A special use of the DISPLAY command is to retrieve saved screen images and redisplay them. Usually a stored screen image is one of the formats produced by the various commands, but it may even be a screen image the user has keyed in.

The display is adapted to the size of the display screen being used including the degenerate case of a typewriter terminal (120 columns by one line).

The first row under the heading rows always has a field name tag, even when it is a continuation of an element value begun on the previous screen.

[illegible]

```
nn          = relative record in set/key aa
mmm etc.   = up to 30 characters of key value.
p, q       = up to 30 characters of element value.
rrr etc.   = 77 character element value.
www etc.   = key field name.
xxx etc.   = field name having a single short element.
yyy etc.   = field name having two short elements.
zzz etc.   = field name having a single long element.
```

COLUMNAR

DISPLAY aa, FFF, ccccc (original command parameters)

PAGE xx

```

----- t1 -----
----- t2 -----
----- : -----
----- : -----
----- tn -----
----- h1 -----
----- h2 -----
----- : -----
----- : -----
----- hn -----

```

```

F1  F2          F5
    F2
      F3  F4
        F4
      F3  F4
      F3

```

where:

xx = page number.
t1 = one or more title lines.
h1 = one or more header lines.
F1,F5 = one element field on the anchor or associate file.
F2 = a multi-element field.
F3,F4 = an elemental field on a subfile.

TOPIC F.5 - DATA RETRIEVAL

A. DATA SET NAME:

PARSED Table

B. CREATED BY:

The SELECT Command (DBSLCT)

C. TYPE OF FILE:

Table

D. ORGANIZATION:

Linear Structure of elements

E. KEY IDENTIFIER (CONTROL FIELD):

Not Applicable

F. RECORD LENGTH:

Not Applicable

G. BLOCKING FACTOR:

Not Applicable

H. PURPOSE:

PARSED is a group of structures used by SELECT to save the information from a parsed expression, when that expression requires a search or contains "S" numbers which cannot be resolved with set numbers until after execution at the actual linear search.

At search time EXSEARCH calls SELECT to begin evaluation of the boolean expressions and to post sets to be searched. SELECT uses the information contained in PARSED to do this and to replace all "S" numbers with their corresponding set numbers. After the search SELECT proceeds with the final evaluation of the expression.

1. PARSED is a based structure consisting of pointers to the other structures containing the various pieces of information that needs to be saved when a boolean expression contains "S" numbers.

2. PARS_TAB_PTR is a pointer to the structure

which is used to describe each element of the expression.

2. PTAB_INFO_PTR is a pointer to the structure which holds additional information about each element of the expression.
 2. PTAB_PTRS_PTR is a pointer to the array of pointers, each pointer corresponding to an element of the expression.
 2. INST_LIST_PTR is a pointer to the list of instructions generated by SELECT to provide for evaluation of the expression.
 2. WAS_PTR is a pointer to the work string in which the expression and other necessary character strings are stored.
 2. LNTN is the length for allocation of all tables listed here except WAS. It is determined from the length of the input expression.
 2. S# contains the S# in which the PARSED pointer is stored, i.e. the PARSED pointer is stored in ENTRYDEF.PARSED pointed to by SRCHTAB.DEFPTR(PARSED.S#).
1. PARS_TAB is the primary table for storage of information as the expression is parsed.
 2. LNTN is the number of array elements in the table.
 2. EL is an element of the table. One element in the table is used to describe each syntactic item in the expression.
 3. IDX is the relative position of the item in the string WAS.
 3. LTH is the length of the item.
 3. ID is the identifier of the item which distinguishes between items of the same general type.
 3. TYPE is the general type of the item, such a relation operator, character string, etc.
 3. TERM is used to mark an item as being a

term during expression evaluation, or to mark an item so that it will be ignored by later passes.

3. SKIP causes later program passes to skip over a particular number of items (or elements in PARS_TAB).
1. PTAB_PTRS is an array of pointers. Each pointer corresponds directly to an element in PARS_TAB; the Nth pointer in PTAB_PTRS corresponds to the Nth element in PARS_TAB. When an expression item results in the formation of a set, the pointer to the set is stored in the corresponding element of PTAB_PTRS.
2. LNTH is number of array elements.
2. EI is a pointer array element.
1. PTAB_INFO is a table for storage of additional information about an expression item, and again each element corresponds directly to an element in PARS_TAB.
2. LNTH is number of array elements.
2. EL is an array element.
3. IDX relative position of item, in string WAS, which is associated with item to which this element corresponds.
3. SFX indicates subfile which applies to item.
3. INDXD on if item (Fieldname or value) is indexed.
3. NNDXE on if item (Fieldname or value) is not indexed.
3. CTL on when item (Fieldname) is control field name.
1. INST_LIST is a list of "instructions" created and executed by SELECT. The instructions guide the creation of sets, both from index files and through linear search, as well as the boolean combination of all sets, once formed, to yield the final set.
2. LNTH number of instruction elements in this

list.

2. EI an instruction.

3. OP is the operation code.

3. IDX1 first parameter/

3. IDX2 second parameter.

3. IDX3 third parameter.

1. WAS is a work string containing the input expression and other necessary character strings.

2. LNTN length of work string.

2. S actual string.

1. WAA is a one-character-per-element array which is defined on top of WAS to allow easy access to a single character.

2. LNTN is number of elements.

2. A is a one character element.

PARSED_LIST is base pointer for PARSED structure.

PARS_TAB_PTR is base pointer for PARS_TAB.

PTAB_INFO_PTR is base pointer for PTAB_INFO.

PTAB_PTRS_PTR is base pointer for PTAB_PTRS.

WAS_PTR is base pointer for WAS and WAA.

INST_LIST_PTR is base pointer for INST_LIST.

WAS_SIZE is set to adjust size of WAS at allocation.

TOPIC F.6 - DATA RETRIEVAL

A. DATA SET NAME:

SETS Display Format

B. CREATED BY:

SETS (DBSETS)

C. TYPE OF FILE:

Terminal Communication

D. ORGANIZATION:

Character Screen Display

E. KEY IDENTIFIER (CONTROL FIELD):

Not Applicable

F. RECORD LENGTH:

320 Bytes typical - 40 column, 8 line output area apart from the prompting area.

G. BLOCKING FACTOR:

Not Applicable

H. PURPOSE:

This is the output created by the SETS command. It is a display on the user's screen or typewriter terminal of the sets created during the current strategy session.

This display consists of the set number or S-number, the number of index references in the set, and the expression (including the control field name, if applicable) that formed the set. The expression will wrap around if it exceeds one line.

Paging forwards and backwards is available. The word 'MORE:' will appear at the bottom of the list if there is more data forward.

I. SAMPLE OUTPUT:

ENTER : SETS

SET#	XREFS	EXPRESSION	PAGE
aa	bbbb	cccccccccccccccccccccccc	1
	"	cccccc	
	"		
	"		
	"		
aa	bbbb	(FROM: ddddddd) cccccc	

-MORE:

Where:

aa = set number,
 bbbb = number of references,
 ccc etc. = expression,
 ddddddd = control field name,
 -MORE: = forward continuation indicator.

TOPIC F.7 - DATA RETRIEVAL

A. DATA SET NAME:

DBINIT - Transient Module Interfaces

B. CREATED BY:

DBINIT

C. TYPE OF FILE:

Table

D. ORGANIZATION:

Documentary Table

E. KEY IDENTIFIER (CONTROL FIELD):

Not Applicable

F. RECORD LENGTH:

Not Applicable

G. BLOCKING FACTOR:

Not Applicable

H. PURPOSE:

The transient module interfaces (see Table 1) are the values assigned to the first parameter of every transient module entry. These values inform DBINIT, the retrieval subsystem director, which modules are to be called next.

Due to the inability in OS for one transient module to call another, the retrieval director has been given the logic to perform all inter-transient module calls. This is accomplished with a half-word (2 bytes) parameter passed to every module called by DBINIT (and the TS supervisor for paging entry points). This parameter has two functions. On entry the value of this parameter dictates the processing flow. Since transient modules have only one entry point, what normally would have been accomplished with additional entry points is provided for with this parameter value on entry. On return, this parameter's value will direct DBINIT to call another module in the table with or without recall or go on and prompt the user

for the next command. Consider these functions separately.

On Entry:

All of the retrieval commands except CANCEL result in DBINIT calling the entry point so indicated in Table 1 with the parameter's value of zero (0). The CANCEL command results in a value of 14 for the entry parameter value.

Since transient modules may only have one entry point, their paging entry point must be the primary module entry point. Thus, the Terminal Support supervisor has adopted a parameter entry value of 100 to indicate paging at the main entry.

On Return:

When a transient module (or any command module called by DBINIT) is finished processing, it must reset the parameter value to zero (0) before returning to prevent DBINIT from calling another module. It is recommended that the parameter be set to zero (0) immediately upon entry to assure its return value in the event of an END or ATTN condition being raised. This is also true for paging entry point processing.

When a transient module (or any command module called by DBINIT) desires to call another transient module, the parameter will have to be set at one of the values in Table 1 to accomplish the call by DBINIT. For example, if a module wished to call the CANCEL SEARCH service entry in DBEXSR, the parameter should be set to 16 and then return. When DBEXSR is then called by DBINIT, its entry parameter value will remain at 16; parameter values remain as returned for any calls done by DBINIT as a result of returning parameter values.

If a module performing an indirect call through DBINIT wishes to be re-called by DBINIT when the subsequent module returns, the returning parameter value should be 100 plus the desired entry value in Table 1. In the example in the previous paragraph, the returning value would be 116. Note that the subsequently called module sees a parameter value of 16 (not 116). On re-call the entry parameter value will be zero (or 14 in the case of CANCEL).

Returning Entry

Parameter Value	Point Called	Function At Entry
3	DBXPND	primary EXPAND command entry
4	DBSLCT0	primary SELECT command entry
5	DBDSPL	primary DISPLAY command entry
6	DBSETS	primary SETS command entry
7	DBFLDS	primary FIELDS command entry
8	DBFORM	primary FORMAT command entry
9	DBSTR2	FORMATS command entry
10	DBSLCT1	SEARCH command entry
11	DBPRNT	primary PRINT command entry
12	DECORR	primary CORRECT command entry
13	DBEXSR	primary CORRECT command entry
14	DBPRNT	CANCEL command entry
15	DBPRNT	S number PRINT service entry
16	DBEXSR	CANCEL SEARCH service entry
17	DBDSPLA	second DISPLAY module entry
18	DBFORMA	second FORMAT module entry
(100		universal system paging value at paging entry)

Table 1. Retrieval Transient Parameter Values

TOPIC F.8 - DATA RETRIEVAL

A. DATA SET NAME:

PRINT Data Set Format

B. CREATED BY:

PRINT (DEPRNT)

C. TYPE OF FILE:

- (5) Non-data base file and
- (2) Formatted print-out

D. ORGANIZATION:

VSAM

E. KEY IDENTIFIER (CONTROL FIELD):

Not Applicable

F. PRINT LENGTH:

132 Bytes maximum printed plus record length and carriage control fields (5 bytes).

G. BLOCKING FACTOR:

Block size = 4096 bytes.

H. PURPOSE:

This is an output data set produced by the PRINT command. It consists of line images written using a PL/I file named PRINTER. At the end of a terminal session an OS PRINT job may be initiated to print the data set off-line on a line printer.

A leader page shows the user's name and mail stop for distribution. Following the output produced for each PRINT command is a separator page having 36 dollar signs on the first line.

PRINT Command - LEADER PAGE

DISTRIBUTE TO: xxxxxxxxxx etc.

MAIL STOP: yyyyyyyy etc.

where:

xxx etc. = user's name

yyy etc. = mail stop

PRINT Command - TYPICAL FORMAT 1 PAGE

PRINT OF SET xx, Format 1.

```
aaaaaaaa: ddddddd
aaaaaaaa: eeeeeee
aaaaaaaa: fffffff
```

where:

```
aaaaaaaa = key field name
d thru f = key value (wraps around to column 1 if more than
            122 characters).
```

PRINT Command - TYPICAL FORMAT 2, 3 or 4 PAGE

PRINT OF SET xx, FORMAT y, zzzzzzzz:vvvvvvvvvvvvvv PAGE wwwww

```

aazaaaaa: d
bbtbbbbb: e
      : f
cccccccc: gggg ..... gggg
gggggggggggggggggg
gggggg

```

where:

```

aazaaaaa = field name having a single short element.
bbtbbbbb = field name having two short elements.
cccccccc = field name having a long element.
d, f      = element value up to 122 characters (no maximum
            number of elements).
ggg etc.  = 379 character element value (no maximum).
zzz etc.  = key field name.
vvv etc.  = first 74 characters of key value.

```

TOEIC F.9 - DATA RETRIEVAL

A. DATA SET NAME:

EXPTAB - Expand Term Table

B. CREATED BY:

DBXPND

C. TYPE OF FILE:

(4) Table

D. ORGANIZATION:

PL/I Data Structure

E. KEY IDENTIFIER (CONTROL FIELD):

Not Applicable

F. RECORD LENGTH:

Not Applicable

G. BLOCKING FACTOR:

Not Applicable

H. PURPOSE:

1. EXPTAB EXTERNAL CONTROLLED.

This table contains a list of alphabetically sequential terms taken from an inverted index file and information relating the terms to reference numbers (E-numbers) used in the SELECT command.

2. TERMS AREA (1250).

This area contains a linked list of terms read from the inverted index.

2. FIRST_PTR POINTER.

Points to the first term in the linked list.

2. LAST_PTR POINTER.

Points to the last term in the linked list.

2. TOP_PTR POINTER

Points to the first term displayed on the current page of data.

- 2. BOTTOM-PTR POINTER
Points to the last term displayed on the current page of data.
- 2. FIRST_E# BINARY FIXED.
Contains the reference number for the first terms in the list.
- 2. LAST_E# BINARY FIXED.
Contains the reference number for the last term in the list.
- 2. TOP_E# BINARY FIXED.
Contains the reference number for the first term on the current page.
- 2. BOTTOM_E# BINARY FIXED.
Contains the reference number for the last term on the current page.
- 2. LO_E# BINARY FIXED.
Contains the lowest valid reference number.
(Either 1 or the reference number of the index origin.)
- 2. HI_E# BINARY FIXED.
Contains the highest valid reference number.
(Either 999 or the reference number of the index end.)
- 2. FLD_NAME CHAR(8) VAR.
Contains the index field name upon which the terms have been expanded.

I. PL/I DECLARATION

```

DC1 1 EXPTAB EXT CONTROLLED, /*DEFINE THE TERM TABLE */
2 TERMS AREA(1250), /*TERM STCRAGE AREA */
2 FIRST_PTR POINTER, /*FIRST LIST ENTRY POINTER*/
2 LAST_PTR PCINTER, /*LAST LIST ENTRY POINTER */
2 TOP_PTR POINTER, /*FIRST LINE CN PAGE PTR */
2 BOTTOM_PTR POINTER, /*LAST LINE ON PAGE PTR */
2 FIRST_E# BIN FIXED, /*FIRST ENTRY'S E# */
2 LAST_E# BIN FIXED, /*LAST ENTRY'S E# */
2 TOP_E# BIN FIXED, /*FIRST E# ON PAGE */
2 BOTTOM_E# BIN FIXED, /*LAST E# ON PAGE */
2 LO_E# BIN FIXED, /*LOWEST VALID E# */
2 HI_E# BIN FIXED, /*HIGHEST VALID E# */
2 FLD_NAME CHAR(8) VAR; /*EXPANDED FIELD NAME */

```

TOPIC F.10 - DATA RETRIEVAL

A. DATA SET NAME:

FLDTAB - Field Name Table

B. CREATED BY:

EBPFLDT entry of module DEPAC

C. TYPE OF FILE:

(4) Table

D. ORGANIZATION:

linear structure containing adjustable arrays.

E. KEY IDENTIFIER (CONTROL FIELD):

FLDTAB is the major structure name. It is the name of an external variable containing the data.

F. RECORD LENGTH:

Not Applicable.

G. BLOCKING FACTOR:

Not Applicable

H. PURPOSE:

The field name table FLDTAB contains the file name and the number of fields available for the user. (The count does not include the FIELDS field.) The sizes, base addresses and names of sequential format definition tables are tabulated. The base addresses and names of columnar format definition tables are tabulated.

I. SCHEMATIC DIAGRAM:

See Figure 1

J. PL/I DECLARATION:

```
/*  FLDTAB:   NASIS   SYSTEM   FIELD   NAME   TABLE   FOR
   DATABASE-2.
```

THIS TABLE IS ALLOCATED (OR FREED AND REALLOCATED) AND INITIALIZED BY A CALL TO THE ENTRY POINT DBPFIDT OF MODULE DEPAK. EACH CALL TO THIS ENTRY POINT CAUSES THE ENTIRE TABLE TO BE RE-INITIALIZED TO THE VALUES FOR THE CURRENTLY SPECIFIED DATABASE FILE. THE VALUES WILL BE ADJUSTED TO REFLECT THE DATA AVAILABILITY BASED UPON THE SECURITY CODE ENTERED BY THE USER. */

DECLARE

```
1  FLDTAB EXT CONTROLLED,          /*NASIS FIELD NAME TABLE */

3  DATAPLEX CHARACTER(8),          /*THE DATABASE FILE NAME */

3  FIELD,                          /*THE DATABASE FID NAMES */
   5 # FIXED BINARY,              /*THE NUMBER OF FIELD    */
                                   /*NAMES IN THE TABLE    */

3  SEQ_FORMAT(25),                /*SEQUENTIAL FORMAT INDEX*/
   5 # FIXED BINARY,              /*THE NUMBER OF FIELD    */
                                   /*NAMES IN THE FORMAT    */
   5 BASE POINTER,                /*THE FORMAT DESCRIPTION */
                                   /*TABLE ADDRESS          */
   5 NAME CHARACTER(8),           /*THE NAME ASSIGNED TO   */
                                   /*THIS FORMAT (OPTIONAL) */

3  COL_FORMAT(25),                /*COLUMNAR FORMAT INDEX */
   5 BASE POINTER,                /*THE FORMAT DESCRIPTION */
                                   /*TABLE ADDRESS          */
   5 NAME CHARACTER(8);           /*THE NAME ASSIGNED TO   */
                                   /*THIS FORMAT (OPTIONAL) */
```

K. FIELD DETAILS:

DATA BASE - has the name of the current dataplex.

FIELD.# - the number of field names excepting RELEN.

SEQ_FORMAT - an array serving as a directory of the sequential format definition tables. The first four entries are posted by RDEPAC to overlay FIELD.NAME beginning with FIELD.KEYNAME as shown in Paragraph I. The remaining entries are posted by RDEFORM to refer to dynamically allocated

sequential format definition tables.

SEQ_FORMAT.# - the number of field names in a sequential format definition table.

SEQ_FORMAT.BASE - the address of a sequential format definition table or a NULL pointer value if it is undefined or the formats 1 through 5.

SEQ_FORMAT.NAME - the name assigned to a sequential format or blanks.

COL_FORMAT - an array serving as a directory of the columnar format definition tables. The entries are posted by RDBFORM to refer to dynamically allocated columnar format definition tables.

COL_FORMAT.BASE - the address of a columnar format definition table or a NULL pointer value if it is undefined.

COL_FORMAT.NAME - the name assigned to a columnar format or blanks.

DATA BASE
NAME

FILE \$\$

NO. OF FIELDS

16

SEQUENTIAL
FORMATS

	NO. OF	BASE ADDR.	NAME
1	1	NULL	
2	5	NULL	
3	8	NULL	
4	16	NULL	
5	8	NULL	
.			
.			
.			
25	5	(ADDR)	

COLUMNAR
FORMATS

	BASE ADDR.	NAME
1	(ADDR)	PAYROLL
2	(ADDR)	CUSTOMER
.		
.		
.		
25		

Figure 1. Schematic Diagram of FLDTAB

TOPIC F.11 - DATA RETRIEVAL

A. DATA SET NAME:

FORMATS Display Format

B. CREATED BY:

Formats - DBSTRT

C. TYPE OF FILE:

Terminal Display (Pageable)

D. ORGANIZATION:

Not Applicable

E. KEY IDENTIFIER (CONTROL FIELD):

Not Applicable

F. RECORD LENGTH:

Not Applicable

G. BLOCKING FACTOR:

Not Applicable

H. PURPOSE:

This terminal display is created to display the names of the formats currently available to him. A title, identifying the display, is generated, followed by the format names. The eight character names are sorted into alphabetic sequence, tagged with an asterisk if the format is in core, separated by a blank and grouped into a SCRNWTH size line before they are written to the display.

TOPIC F.12 - DATA RETRIEVAL

A. DATA SET NAME:

SETAB Sets Table

B. CREATED BY:

DBINIT and modified by DBSETU

C. TYPE OF FILE:

Table

D. ORGANIZATION:

PL/I Data Structure

E. KEY IDENTIFIER (CONTROL FIELD):

Not Applicable

F. RECORD LENGTH:

Not Applicable

G. BLOCKING FACTOR:

Not Applicable

H. PURPOSE:

1. SETAB EXTERNAL CONTROLLED.
This structure contains the sets, i.e., current strategy, that the user is creating and associated information.
2. CURRENT_# BINARY FIXED (15,0).
This is the value of the last set number that was created.
2. SET (0:99).
There is one set created for each select, search and LIMIT COMMAND.
3. POINTER POINTER.
There is a pointer to a list of keys for every set that is created. POINTER (J) points to the list for SET (J).
3. SIZE BINARY FIXED (31,0).
This is the number of keys associated with

the corresponding set number.

3. TYPE CHARACTER (1).

This is the SUBFILE SUFFIX that describes the origin of the keys in the set.

I. PL/I DECLARATION:

```

/*                NASIS SYSTEM SET TABLE                */
DCL 1 SETAB EXTRNL CONTROLLED, /*DEFINE THE SET TABLE  */
  2 CURRENT # BIN FIXED(15,0), /*LAST ASSIGNED SET NUMBER */
  2 SET(0:99),                /*DEFINE THE SET ENTRIES  */
  3 POINTER PTR,              /*THE SET LIST PCINTER    */
  3 SIZE BIN FIXED(15,0),     /*THE SET SIZE (# OF KEYS) */
  3 TYPE CHAR(1);             /*THE SET TYPE (SUBFILE ID) */

```

TCHIC F.13 - DATA RETRIEVAL

A. DATA SET NAME:

USERTAB User Data Table

B. CREATED BY:

EDBMTT

C. TYPE OF FILE:

Table

D. ORGANIZATION:

PL/I Data Structure

E. KEY IDENTIFIER (CONTROL FIELD):

Not Applicable

F. RECORD LENGTH:

Not Applicable

G. BLOCKING FACTOR:

Not Applicable

H. PURPOSE:

1. USERTAB EXTERNAL CONTROLLED.

This structure contains user oriented and status information useful to all NASIS sub-systems.

2. NASIS_ID CHARACTER(8) VARYING.

This field contains the id specified by the user when initiating his NASIS session.

2. SECURITY CHARACTER (8) VARYING.

This field contains the user's most recently specified security code, i.e. his PASSWORD at logon or in respond to a SECURE Command.

2. OWNER_ID CHARACTER (8) VARYING.

This field contains the ID associated with the owner of the file specified by the user.

2. STRATEGY CHARACTER (16) VARYING.

This field contains the name of the strategy

in the event of a RERUN.

2. TASK_ID BINARY FIXED (31,0).
This field contains the task identification number assigned to the user at logon time.
2. SEQUENCE BINARY FIXED (15,0).
This field contains a sequence number used by the system in defining unique ddnames to dynamically specified files.
2. BITS.
The status of the user's session is reflected by the settings of the following bit switches.
 3. MTTFLAG BIT(1).
Describes whether the task is running under MTT or not.
 3. DISABLED BIT(1).
Defines the status of attention interrupts.
 3. RETRIEVR BIT(1).
Describes whether the task is running under the retrieval system or not.
 3. RESTART BIT(1).
Describes whether the session is a restart.
 3. RERUN BIT(1).
Describes whether the session is a rerun.
 3. TESTMODE BIT(1).
Describes whether the session is productive or a debugging run.
 3. CONVFLAG BIT(1).
Describes whether the task is conversational or not.
 3. RECALL BIT (1).
Describes whether the last program is to be recalled.

I. PL/I DECLARATION:

```
/*          NASIS SYSTEM USER DATA TABLE          */
```

DCL

```

1 USERTAB EXT CONTROLLED,      /*NASIS USER DATA TAELE      */
2 NASIS_ID CHAR(8) VAR,        /*USER'S IDENTIFICATION        */
2 SECURITY CHAR(8) VAR,        /*USER'S SECURITY CODE         */
2 COWNER_ID CHAR(8) VAR,       /*FILE COWNER'S IDENTIFIER     */
2 STRATEGY CHAR(16) VAR,       /*STRATEGY NAME FOR RERUN      */
2 TASK_ID BIN FIXED(31,0),     /*TASK IDENTIFICATION #        */
2 SEQUENCE BIN FIXED(15,0),    /*DDNAME SEQUENCE NUMBER      */
2 BITS,                        /*SYSTEM STATUS FLAGS         */
3 MTTFLAG BIT(1),             /*'1'=IN MTT MODE             */
3 DISABLED BIT(1),            /*'1'=ATTN'S DISABLED         */
3 RETRIEVE BIT(1),            /*'1'=RUNNING RETRIEVAL       */
3 RESTART BIT(1),             /*'1'=IN RESTART MODE         */
3 RERUN BIT(1),               /*'1'=IN RERUN MODE           */
3 TESTMODE BIT(1),            /*'1'=NO STRATEGY SAVING      */
3 CONVFLAG BIT(1),            /*'1'=CONVERSATIONAL          */
3 RECALL BIT(1);              /*'1'=RECALL LAST PROGRAM     */

```

TOPIC F.14 - DATA RETRIEVAL

A. DATA SET NAME:

EXPLAIN Display Format

B. CREATED BY:

EXPLAIN (message, RESPONSE and term options) - DBEXPL

C. TYPE OF FILE:

Terminal Display (Pageable)

D. ORGANIZATION:

Not Applicable

E. KEY IDENTIFIER (CONTROL FIELD):

Not Applicable

F. RECORD LENGTH:

Not Applicable

G. BLOCKING FACTOR:

Not Applicable

H. PURPOSE:

This terminal display is created to display to the user the results of his message response or term explanation. The data will be written to the screen as read from the message file with no indentation or data tagging, but with word-break specified.

TOPIC F.15 - DATA RETRIEVAL

A. DATA SET NAME:

SPRNTAB - S# PRINT-parameter table

B. CREATED BY:

SELECT (search option) - DESLCT

C. TYPE OF FILE:

Table

D. ORGANIZATION:

Linear structure.

E. KEY IDENTIFIER (CONTROL FIELD):

SPRNTAB is the major structure name; it is the name of the control section containing the data.

F. RECORD LENGTH:

Not Applicable

G. BLOCKING FACTOR:

Not Applicable

H. PURPOSE:

This table contains a set of PRINTparameters saved at the time a user issued a PRINT on an S#. Since S#'s do not refer to a specific list of keys, the actual PRINT is not acted upon until after the search execution. At that time DEEXSR calls DBPRNT to perform the PRINT wherein the SPRNTAB is referenced for the PRINT parameters.

I. SCHEMATIC DIAGRAM:

See Figure 1 of the SRCHTAB Data Set Specification.

J. VARIABLE DETAILS:

1. SPRNTAB is a table of record parameters of an S#.

3. FORMAT is a table of record format parameters.

5. TYPE

5. FIRST

5. LAST

3. NEXT_SPRNTAB is a pointer to next SPRNTAB structure.

TOPIC F.16 - DATA RETRIEVAL

A. DATA SET NAME:

SEQ_FORM - Sequential Format Definition Table

B. CREATED BY:

DBPFLEDT entry DBPAC (formats 1-5)

DBFORM (formats 6-25) - by the FORMAT command.

C. TYPE OF FILE:

(4) Table

D. ORGANIZATION:

Adjustable linear array of 8-character field names.

E. KEY IDENTIFIER (CONTROL FIELD):

SEQ_FORM is the major structure name.

F. RECORD LENGTH:

Not Applicable

G. BLOCKING FACTOR:

Not Applicable

H. PURPOSE:

A sequential format definition table, SEQ_FORM, contains a list of the names of the fields in a sequential format for use by the DISPLAY and PRINT commands. The key field name is always the first in the list. The number of names in the list and the base address of the list is posted in FLDTAB.SEQ_FORMAT.#.

I. PL/I DECLARATION:

```
DECLARE
  1 SEQ_FORM BASED (SEQ_BASE), /* SEQUENTIAL FORMAT SPECS */
  3 FIELD_1_, /* OVERLAID BY FIELD(1) */
  5 #FIELDS FIXED BIN, /* NOT USED */
  5 PAD CHAR(6), /* FILLER TO CHAR(8) */
  3 FIELD(2:I
  REPER (SEQ_FORM.#FIELDS)), /* NOT USED */
  5 NAME CHAR(8); /* NAME LIST */
```

TOPIC F.17 - DATA RETRIEVAL

A. DATA SET NAME:

VALUTAB - a linear search table of test values

B. CREATED BY:

SELECT (search option) - DBSLCT

C. TYPE OF FILE:

Table

D. ORGANIZATION:

Linear structure.

E. KEY IDENTIFIER (CONTROL FIELD):

VALUTAB is the major structure name; it is the name of the control section containing the data.

F. RECORD LENGTH:

Not Applicable

G. BLOCKING FACTOR:

Not Applicable

H. PURPOSE:

This table contains the number of and the pointers to the test values associated with an S#. Each pointer points to a based structure containing the value length and the value string, called VALUE.

I. SCHEMATIC DIAGRAM:

See Figure 1 of the SRCHTAE Data Set Specification.

J. VARIABLE DETAILS:

1. VALU_# is set to adjust the size of the VALUPTR array in VALUTAB.

1. VALUTAB is a table of pointers to values.

3. #OF is number of pointers in this table.

3. VALUPTR is an array of pointers to values.

1. VALUE_SIZE is set for size of value at allocation.
1. VALUE is a table containing a value to be used during search. Pointer is in VALUTAB.
3. SIZE is length of value.
3. X is actual value.

TOEIC F.18 - DATA RETRIEVAL

A. DATA SET NAME:

SRCHTAB - Linear Search Table of Pseudo-sets

B. CREATED BY:

SELECT (search option) - DBSLCT

C. TYPE OF FILE:

Table

D. ORGANIZATION:

Linear structure containing arrays.

E. KEY IDENTIFIER (CONTROL FIELD):

SRCHTAB is the major structure name; it is the name of the control section containing the data.

F. RECORD LENGTH:

Not Applicable.

G. BLOCKING FACTOR:

Not Applicable

H. PURPOSE:

The linear search table of pseudo-sets, SRCHTAB, contains pointers to the ENTRYDEF structure allocated for each defined S#, whether a PRINT is to be performed on any S#, and various control switches used during the search execution between DBEXSP and DBSLCT interactively.

I. SCHEMATIC DIAGRAM:

See Figure 1

J. VARIABLE DETAILS:

1. MAX_S is the maximum allowable pseudo-set number.

1. SRCHTAB is the main search table.

3. CURRENT_S# is the last pseudo-set number

assigned.

3. SRCH_IN_PROGRESS is a bit on if a search is being executed.
3. IFSC is set on if any pseudo-set is to be printed.
3. SLCTTRN is set on by DBEXSR before a transient call to DESLCT as a return flag.
3. PRNTRN is set on by DBEXSR before a transient call to DBPRNT as a return flag.
3. SLCT_ERROR is a bit on if error occurs in SELECT during execution of search.
3. SLCT_FINISH is a bit on if all SELECT functions are complete during execution of search.
3. DEFPTR is an array of pointers which point to a ENTRYDEF structure for each defined S#.

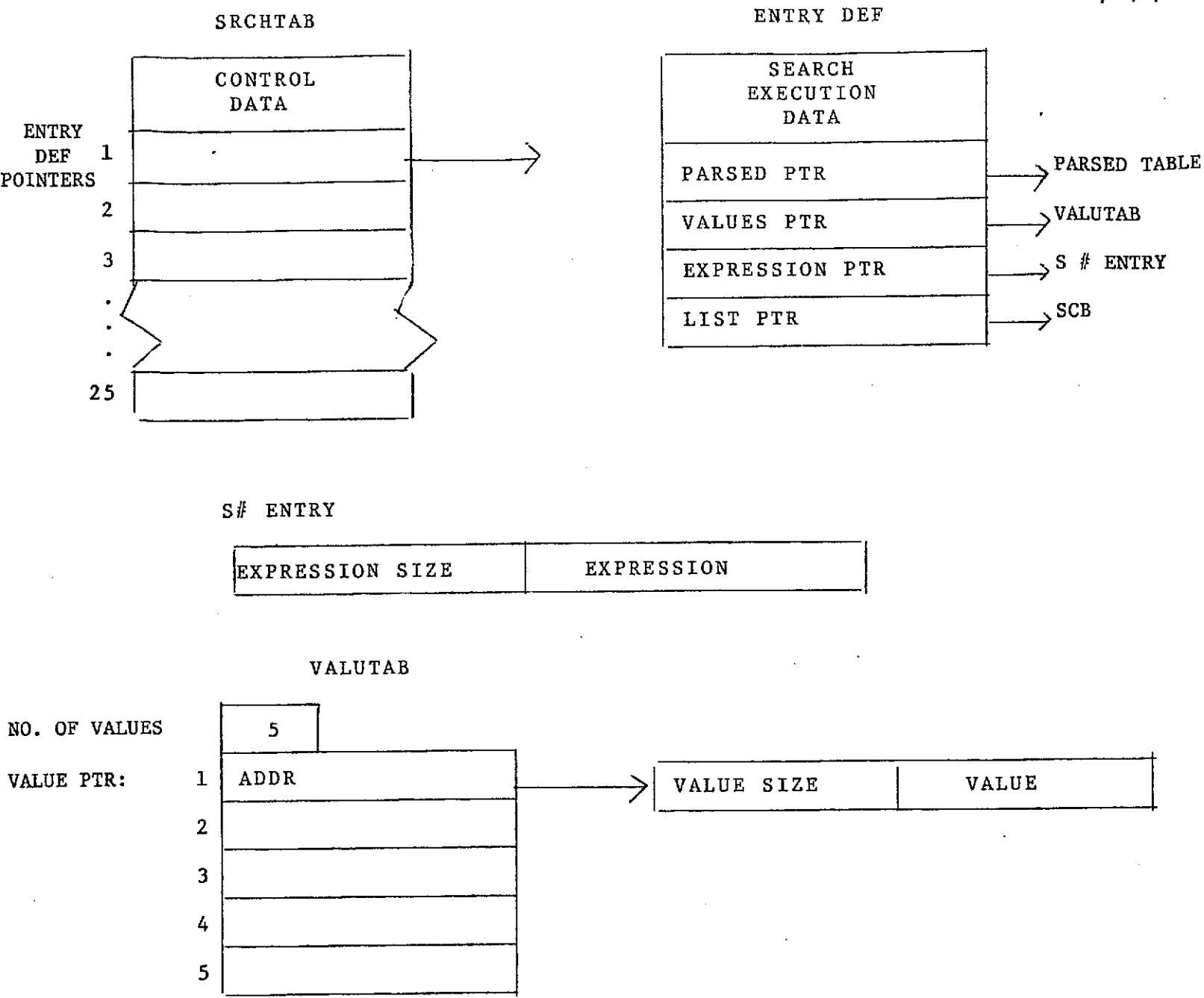


Figure 1. Schematic Diagram of Search Tables

TOEIC P.19 - DATA RETRIEVAL

A. DATA SET NAME:

COL_FORM - Columnar Format Definition Table

B. CREATED BY:

DBFORM - the FORMAT command

C. TYPE OF FILE:

(4) Table

D. ORGANIZATION:

Structure containing miscellaneous items, a linear array, and an adjustable array of structures.

E. KEY IDENTIFIER (CONTROL FIELD):

COL_FORM is the major structure name.

F. RECORD LENGTH:

Not Applicable.

G. BLOCKING FACTOR:

Not Applicable

H. PURPOSE:

A columnar format definition table, COL_FORM, contains coded specifications for a columnar display. It is used by the DISPLAY and PRINT commands and may be revised by the FORMAT command. The optional line for the page number, lines for titles, and lines for headers hold literal text for output. For each field specified, the field name, column, width, summary requirements, tally, and summation values are carried. (Average is not carried in COL_FORM; it is computed in DISPLAY or PRINT.)

I. PL/I DECLARATION:

```

DECLARE
1 CCL_FORM BASED(COL_BASE),    /*COLUMNAR FORMAT SPECS  */
3 LINESIZE FIXED BIN(31),      /*SCRNCCI OR 132          */
3 RECORD_COUNT FIXED BIN(31), /*INIT(0)                 */
3 TOP,
5 (PAGE#,                      /*1 OR 0 LINES            */

```

```

      #TITLES,                /*0 OF MORE LINES      */
      #HEADERS,               /*0 CF MORE LINES     */
      DEFAULT_HDR) FIXED BIN, /*C OR RELATIVE HEADER LINE*/
5 LINE(10) CHAR(132),
3 COL_GIVEN BIT(1),          /*1: FIELD COLUMNS GIVEN */
                              /*0: FIELD COLUMNS DEFALTD*/

3 #FIELDS FIXED BIN,
3 FIELD( I REFER(COL_FORM.#FIELDS)),
5 NAME CHAR(8),
5 COLUMN FIXED BIN,          /*FOR TRUNCATION INDICATOR*/
                              /*USE COLUMN+1...FOR VALUE*/
5 WIDTH FIXED BIN,           /*WITHOUT              */
                              /*TRUNCATION INDICATOR  */
5 ELEMENT_LIMIT FIXED BIN, /*FOR RETRIEVAL        */
5 ELEMENT_TALLY,
7 REQUIRED BIT(1),            /*INIT('0'B)          */
7 # FIXED BIN(31),           /*INIT(0)              */
5 ELEMENT_SUM,
7 REQUIRED BIT(1),            /*INIT('0'B)          */
7 ZONED BIT(1),              /*1: ZONED VALUE (INIT) */
                              /*0: BINARY VALUE       */
7 VALUE FLOAT BIN(53),       /*INIT(0)              */
5 ELEMENT_AVERAGE_REQUIRED
BIT(1);                      /*INIT('0'B)          */

```

TOPIC F.20 - DATA RETRIEVAL

A. DATA SET NAME:

FIELDS Display Format

B. CREATED BY:

FIELDS - DBFIDS

C. TYPE OF FILE:

Terminal Display (Pageable)

D. ORGANIZATION:

Not Applicable

E. KEY IDENTIFIER (CONTROL FIELD):

Not Applicable

F. RECORD LENGTH:

Not Applicable

G. BLOCKING FACTOR:

Not Applicable

H. PURPOSE:

This terminal display is created to display the names of the fields available to him from the current file. A title, identifying the display, is generated, followed by the field names. The eight character names, flagged by an asterisk, if indexed, and separated by a blank, are grouped into SCRNWTH size lines before they are written to the display. As each subfile is encountered, a heading, identifying it and its control field, is generated.

TOEIC F.21 - DATA RETRIEVAL

A. DATA SET NAME:

S#ENTRY - S# Expression Data Table

B. CREATED BY:

SELECT (search option) - DBSLCT

C. TYPE OF FILE:

Table

D. ORGANIZATION:

Linear structure.

E. KEY IDENTIFIER (CONTROL FIELD):

S#ENTRY is the major structure name; it is the name of the control section containing the data.

F. RECORD LENGTH:

Not Applicable

G. BLOCKING FACTOR:

Not Applicable

H. PURPOSE:

This table contains the actual screen image expression for each defined S#. This table is allocated by DBSLCT and referenced by DBEXSR and DBSETS.

I. SCHEMATIC DIAGRAM:

See Figure 1 of the SRCHTAP Data Set Specification.

J. VARIABLE DETAILS:

1. S#ENTRY is a structure of S# expression displayable information.

3. EXPRSN_SIZE is the allocated size of the variable part of the expression for this S#.

3. The remaining variables contiguously represent the S# expression.

TOPIC F.22 - DATA RETRIEVAL

A. DATA SET NAME:

ENTRYDEF table of S# information for linear search

B. CREATED BY:

SELECT (search option) - DBSLCT

C. TYPE OF FILE:

Table

D. ORGANIZATION:

Linear structure.

E. KEY IDENTIFIER (CONTROL FIELD):

ENTRYDEF is the major structure name; it is the name of the control section containing the data.

F. RECORD LENGTH:

Not Applicable

G. BLOCKING FACTOR:

Not Applicable

H. PURPOSE:

This table contains associated information per S# posted at definition time by DBSLCT and during the search execution by DBEXSR. Contained within ENTRYDEF are pointers to the SPRNTAB structure of PRINT parameters for an S#, to the temporary list keys, to the parsing information, to the VALUTAB structure of test values, and the S# expression in S#ENTRY.

I. SCHEMATIC DIAGRAM:

See Figure 1 of the SRCHTAB Data Set Specification.

J. VARIABLE DETAILS:

1 EXPRSN_LEN is the dynamic length of a particular S# entry's expression; see S#ENTRY.

1 ENTRYDEF is an array with detailed pseudo-set information:

- 5 DELETED is a bit on if this pseudo-set has been deleted.
- 5 ACTIVE is set on by DBEXSR during the search execution if the S# is involved in the current search pass.
- 5 RECORD is a pointer used for the parameters of those pseudo-sets to be RECORDED after a search execution. This points to SPRNTAB structure.
- 5 CREATED_BY is two bits identifying the type of SELECT command used to create this pseudo-set where,
 - 7 SELECT_IF bit is on if the search option was used, or
 - 7 SELECT_BOOI bit is on if the boolean option was used.
- 5 REF_SET is a structure used for identifying the searching universe (or set) wherein,
 - 7 PTR is pointer to set to search within
 - 7 S# is a bit on if the set to be searched is a pseudo-set.
- 5 CORRES_SET# is the value of the set resulting from this pseudo-set.
- 5 LIST_PNTR is a pointer to the search list structure for this pseudo-set.
- 5 PARSED is a pointer to a parsing structure for boolean-created pseudo-sets.
- 5 FIELDNAM is the field name to be tested.
- 5 OP_CODE is a value of the operator to be used for the test, as follows:
 - 1. greater than
 - 2. less than
 - 3. equal
 - 4. greater than or equal
 - 5. less than or equal
 - 6. not equal
 - 7. between
 - 8. containing

- 5 VALUES is a pointer to test values; this points to VALUTAB.
- 5. EXFF_PTR is a pointer to the S#ENTRY structure.

TOHIC G.1 - USAGE STATISTICS

A. DATA SET NAME:

STATIC Data Set Descriptors

B. CREATED BY:

Command System and Maintenance System

C. TYPE OF FILE:

Dataplex

D. ORGANIZATION:

ISAM

E. KEY IDENTIFIER (CONTROL FIELD):

See PURPOSE discussion of KEY.

F. RECORD LENGTH:

4000/V

G. BLOCKING FACTOR:

Not Applicable

H. PURPOSE:

Maintain system statistics:

1. Retention of Statistics

In order to maintain usage statistics a file is required. To do so efficiently the file will be a simple ISAM data set. It has been shown that undue overhead of DBPL/1 causes modules accessing data using these facilities to run 5 to 10 times longer. Since the gathering of statistics should ideally not be reflected in the gathered statistics a simple file is to be use. The standard name for the file is to be NASIS.STATIC

2. Accumulation of Statistics

The STATIC file will be composed of two different record types. The data required, and how it will be kept, is as follows:

a. KEY

SEPARATE - A single character which will distinguish the record type. A value of zero will indicate a maintenance record. A value of one will indicate a retrieval record.

IDENTIFIER - For maintenance records, it will be the data base name padded with dollar signs. For retrieval records, it will be the NASIS-ID padded with asterisks (to eight characters). Appended to the NASIS-ID will be the data base owner's ID, the file where this field exists, and the field name.

b. The maintenance fields are as follows:

TOTALTRN - the number of transactions processed.

ANCOUNT - the number of records on the anchor file.

TOTALRUN - the number of maintenance runs.

MAINDATE - the date of each maintenance run: element 1 will be the dates of the data base creation, elements 2 through 13 will be the dates of individual maintenance runs. After the dates have been filled, the second one will be dropped and the newest date added on the end. This field is treated as a 13 element char (6) array.

TRANCNEW

TRANCDEL

TRANCUPD

TRSUBNEW

TRSUBDEL

TRSUBUPD

TRINVNEW

TRINVDEL

TRINVUPD

- where TR indicates transaction;
ANC, the anchor file; SUB, the
subrecord files; INV, inverted
files; NEW, new records; DEL,
deletions; and UPD, updates.

These are the transaction count fields required for maintenance statistics. These fields will be used in conjunction with the data field. These fields are treated as 13 element fixed binary (31 arrays. The elements will correspond directly with the date field and will represent the number of that given type of transaction encountered during the maintenance run. When all of the elements are present, the next count inserted will cause the second count to be added to the first element and the second element dropped. The newest element will go on the end.

c. The retrieval fields are:

CONNTIME - the connect time

CPUTIME - the CPU time

TOTALES - number of sessions

STRATLEN - the strategy length

STRATSTR - the number of strategies stored

STARDTE - the date of the first terminal
session

LASTDATE - the date of the last terminal
session

NOTE: The eight fields above are to be accumulated for each NASIS-ID. There may be many records for each NASIS-ID; therefore, these statistics will be kept in a special record. The COWNER-ID and the inverted file name in this special record will be equal to blanks.

#EXPANDS - number of EXPANDS per session

#SELECTS - number of SELECTs per session
#SEARCHS - number of SEARCHes per session
#CORECTS - number of CORRECTs per session
SESSDATE - the date of each session

These fields are all treated as 13 element arrays. The first element represents an accumulator and contains the total for all occurrences up to the SESSDATE, which is the date of the last ejected session of the list (the earliest session). Regardless of the actual number of sessions within one calendar day, the statistics will be accumulated as if there were only one session.

All of the maintenance statistics will be automatically updated with the Load/Create program and the Maintenance Mainline program. If the data base owner wishes to modify certain data pertaining to the maintenance statistics, he has the ability to use the CORECT command to update the STATIC data base interactively.

All of the retrieval statistics will be automatically updated with the FINISH module of the command system. If required, at maintenance time, a 'snapshot' of the statistics will be printed. If the data base owner (system manager) wishes to modify certain data pertaining to the retrieval statistics, he has the ability to use the CORRECT command to interactively update the STATIC data base.

APPENDIX A.

The STATIC file is composed of the following fields:

A. KEY

1. Alphanumeric.
2. Fixed field.
3. Length of 32 bytes.
 - a. First byte is maintenance or retrieval record indicator.
 1. 0 = maintenance record.
 - a. data base name left justified.
 - b. remainder padded with '\$'s.
 2. 1 = retrieval record.
 - a. NASIS-ID//OWNER-ID//file of data base//field name.
 1. The NASIS-ID is eight characters long and padded with '*'s.
 2. The OWNER-ID is really a TSS-ID, eight characters long and padded with '*'s.
 3. The file name is that file on which the field for statistics is being gathered.
 4. The field name is the name of field on which statistics are being gathered.

B. TOTALTRN (Maintenance)

1. Alphanumeric
2. Fixed field
3. Length of 6 bytes
4. Contains the total number of transactions.

C. ANCOUNT (Maintenance)

1. Alphanumeric
2. Fixed field.
3. Length of 6 bytes.
4. Contains number of records on the anchor file.

D. TOTALRUN (Maintenance)

1. Alphanumeric.
2. Fixed field.
3. Length of 3 bytes.
4. Contains the number of maintenance runs.

E. MAINDATE (Maintenance)

1. Alphanumeric.
2. Fixed element.
 - a. Total of 13 elements, each 6 bytes long.

- b. In the form MM/DD/YY to indicate the month, day, and year of each maintenance run. Element 1 will contain the data base creation date while elements 2-13 will be the dates of the individual maintenance runs. After the dates have been filled, the second one will be dropped and the newest date added to the end.

- 3. Total length of 78 bytes.

F. TRANCNEW (Maintenance)

- 1. Alphanumeric.
- 2. Fixed elements.
 - a. Each 4 bytes long.
 - b. 13 elements.
- 3. Total length of 52 bytes.

G. TRANCDEL (Maintenance)

- 1. Alphanumeric.
- 2. Fixed element.
 - a. Each 4 bytes long.
 - b. 13 elements.
- 3. Total length of 52 bytes.

H. TRANCUPD (Maintenance)

- 1. Alphanumeric.
- 2. Fixed element.
 - a. Each 4 bytes long.
 - b. 13 elements.
- 3. Total length of 52 bytes.

I. TRSUBNEW (Maintenance)

- 1. Alphanumeric.
- 2. Fixed element.
 - a. Each 4 bytes long.
 - b. 13 elements.
- 3. Total length of 52 bytes.

J. TRSUBDEL (Maintenance)

- 1. Alphanumeric.
- 2. Fixed element.
 - a. Each 4 bytes long.
 - b. 13 elements.
- 3. Total length of 52 bytes.

K. TRSUBUPD (Maintenance)

- 1. Alphanumeric.
- 2. Fixed element.
 - a. Each 4 bytes long.
 - b. 13 elements.
- 3. Total length of 52 bytes.

L. TRINVNEW (Maintenance)

1. Alphanureric.
2. Fixed element.
 - a. Each 4 bytes long.
 - b. 13 elements.
3. Total length of 52 bytes.

M. TRINVDEL (Maintenance)

1. Alphanumeric.
2. Fixed element.
 - a. Each 4 bytes long.
 - b. 13 elements.
3. Total length of 52 bytes.

N. TRINVUPD (Maintenance)

1. Alphanumeric.
2. Fixed element.
 - a. Each 4 bytes long.
 - b. 13 elements.
3. Total length of 52 bytes.

NOTE: Items F through N are transaction count fields for the maintenance statistics and correspond directly to MAINDATE.

TR indicates TRANSACTION
 ANC indicates ANCHOR FILE
 INV indicates INVERTED FILE
 NEW indicates NEW RECORDS
 DEL indicates DELETIONS
 UPD indicates UPDATES

O. CONNTIME (Retrieval)

1. Alphanumeric.
2. Fixed field.
3. Length of 10 bytes.
4. Contains connect time.

P. CPUTIME (Retrieval)

1. Alphanumeric.
2. Fixed field.
3. Length of 10 bytes.
4. Contains total CPU time.

Q. TOTALSES (Retrieval)

1. Alphanumeric.
2. Fixed field.
3. Length of 4 bytes.
4. Contains total number of sessions.

R. STRATLEN (Retrieval)

1. Alphanumeric.
2. Fixed field.
3. Length of 4 bytes.

- 4. Contains strategy length.
- S. STRATSTR (Retrieval)
 - 1. Alphanumeric.
 - 2. Fixed field.
 - 3. Length of 4 bytes.
 - 4. Contains number of strategies stored.
- U. STARTDTE (Retrieval)
 - 1. Alphanumeric.
 - 2. Fixed field.
 - 3. Length of 6 bytes.
 - 4. Contains date of first terminal session.
- V. LASTDATE (Retrieval)
 - 1. Alphanumeric.
 - 2. Fixed field.
 - 3. Length of 6 bytes.

NOTE: The right fields above are accumulated for each NASIS-ID. The owner-ID and the file-name have no meaning.

Therefore, the KEY of the record where these statistics are meaningful will be composed of an owner-ID and a file-name which are blank.

- W. #EXPANDS (Retrieval)
 - 1. Alphanumeric.
 - 2. Fixed element.
 - a. Each 4 bytes long.
 - b. 13 elements.
 - 3. Total length of 52 bytes.
- X. #SELECTS (Retrieval)
 - 1. Alphanumeric.
 - 2. Fixed element.
 - a. Each 4 bytes long.
 - b. 13 elements.
 - 3. Total length of 52 bytes.
- Y. #SEARCHS (Retrieval)
 - 1. Alphanumerics.
 - 2. Fixed element.
 - a. Each 4 bytes long.
 - b. 13 elements.
 - 3. Total length of 52 bytes.
- Z. #CORECTS (Retrieval)
 - 1. Alphanumerics.
 - 2. Fixed element.
 - a. Each 4 bytes long.
 - b. 13 elements.

3. Total length of 52 SESSDATE (Retrieval)

AA. SESSDATE (Retrieval)

1. Alphanumerics.
2. Fixed element.
 - a. Each 6 bytes long.
 - b. Maximum of 13 elements.
3. Total length of 78 bytes.

NOTE: In the last 5 fields there is a one for one correspondence in the elements.

first SESSDATE - the date of the newest session in the accumulated counts.

first (others) - the accumulated counts on all indicated.

Regardless of the actual number of sessions within one given calendar day, the statistics will be accumulated as if there were only one session.

When (during UPDATE) a record is encountered with the variable fields having all 13 elements filled, the 'snapshot' of the given record will be taken. The last 12 elements will then be cleared, by summing them and adding them to the first element, the first element SESSDATE will be made equal to the last element SESSDATE.

TOHC G.2 - USAGE STATISTICS

A. DATA SET NAME:

Maintenance Statistics Report Format

B. CREATED BY:

STATIC Report (NDBPRNTM)

C. TYPE OF FILE:

VS (print)

D. ORGANIZATION:

Sequential

E. KEY IDENTIFIER (CONTROL FIELD):

Not Applicable

F. RECORD LENGTH:

133 Bytes

G. BLOCKING FACTOR:

Not Applicable

H. PURPOSE:

To display (via listing) the status of the maintenance statistics.

MAINTENANCE STATISTICS FOR SYSTEMS MANAGER **

01/11/73

PAGE 1

DATAPLEX NAME	TOTAL TRNS	ANCHOR RECORDS	NUMBER RUNS	TRANS RUN	MAINTENANCE DATES	FILEPLEX			SUBPLEX			XPLEX		
						ADDS	DELETES	UPDATES	ADDS	DELETES	UPDATES	ADDS	DELETES	UPDATES
ASRD1\$		3,132	1		12/19/72	3,132								

FILEPLEX	ADDS	DELETES	UPDATES
----------	------	---------	---------

TOTAL	3,132		
-------	-------	--	--

FOR ALL RUNS

AVERAGE	3,132		
---------	-------	--	--

PER RUN

189

TOEIC G.3 - USAGE STATISTICS

A. DATA SET NAME:

Retrieval Statistics Report Format

B. CREATED BY:

Report Print (NDBPRNTE)

C. TYPE OF FILE:

PS (print)

D. ORGANIZATION:

Sequential

E. KEY IDENTIFIER (CONTROL FIELD):

Not Applicable

F. RECORD LENGTH:

133 Bytes

G. BLOCKING FACTOR:

Not Applicable

H. PURPOSE:

To display (via listing) the status of the retrieval statistics.

RETRIEVAL STATISTICS

01/03/73

NASISID	CONN-TIME HR:MM:SC	CPU-TIME HR:MM:SC:MS	# SES	STRAT LENGTH	STORED #	OWNER ID	FILE NAME	FIELD NAME	ACTUAL EXP	TOTAL SEL	NUMBER OF SRCH	OF CORR
NE01	0:53:30	0:00:48:790	5	0	0							
						SAOWNER	ASRD1\$A	AUTHOR	3	0	0	0
						SAOWNER	ASRD1\$B	KEYWORDS	13	0	0	0
						SAOWNER	DB2TDBA	EMPAGE	1	0	0	0
						SAOWNER	DB2TDBB	TOTALCAR	1	0	0	0
						SAOWNER	DB2TDBC	KIDAGE	1	0	0	0
						SAOWNER	DB2TDBD	PET	1	0	0	0
						SAOWNER	DB2TDBE	SVCDATE	1	0	0	0

TOPIC G.4 - USAGE STATISTICS

A. DATA SET NAME:

Snapshot Statistics Report Format

B. CREATED BY:

Snapshot Print (NDBCHKPT)

C. TYPE OF FILE:

PS (print)

D. ORGANIZATION:

Sequential

E. KEY IDENTIFIER (CONTROL FIELD):

Not Applicable

F. RECORD LENGTH:

133 Bytes

G. BLOCKING FACTOR:

Not Applicable

H. PURPOSE:

To display (via listing) those records which have undergone the reinitialization process.

SNAPSHOT (CHECKPOINT) OF RETRIEVAL STATISTICS RECORDS BEFORE REINITIALIZATION

12/18/72

PAGE 1

LISR ID	CONN-TIME HR:MIN:SC	CPU-TIME HR:MN:SC:MS	# SES	STRAT LENGTH	OWNER-ID	FIELD NAME	FILE NAME	SESSION DATE	# EXPANDS	# SELECTS	# SEARCHS	# CORRECTS
NEO1	:19:40	0:00:12:399	2		SAOWNER	KEYWORDS	ASRDI\$B	721215				
								721215	1			
								721215				
								721215				
								721215				
								721215				
								721215				
								721215				
								721215				
								721215				
								721215				
								721215				
								721215	1			

193

TOHIC H.1 - IMMEDIATE COMMANDS

A. DATA SET NAME:

NASIS Message File - NASIS.MESSAGES

B. CREATED BY:

Not Applicable

C. TYPE OF FILE:

ISAM

D. ORGANIZATION:

Indexed-Sequential

E. KEY IDENTIFIER (CONTROL FIELD):

The fifteen byte key is composed of the eight byte message key concatenated to the seven byte line number.

F. RECORD LENGTH:

F(160), key length is 15 with an RKP = 0.

G. BLOCKING FACTOR:

Not Applicable

H. PURPOSE:

This data set contains the NASIS system messages used by the various modules for prompting and diagnostic messages.

TOPIC H.2 - STRATEGY FILE

A. DATA SET NAME:

NASIS.STRATEGY

B. CREATED BY:

NTSTRAT

C. TYPE OF FILE:

ISAM

D. ORGANIZATION:

Random. File consists of approximately 4000 records. The first 256 contain bit switches flagging which of the remainder are active. The next 256 are base records pointing to the first of a string of chained data records.

E. KEY IDENTIFIER (CONTROL FIELD):

Strategy Name (16 characters)

F. RECORD LENGTH:

150 characters

G. BLOCKING FACTOR:

Not Applicable

H. PURPOSE:

To store strategies, formats and print information created by a NASIS user.

TOPIC H.3 - IMMEDIATE COMMANDS

A. DATA SET NAME:

Strategy Display Format

B. CREATED BY:

DBSTRT

C. TYPE OF FILE:

Screen Display

D. ORGANIZATION:

Header = STRATEGY name (centered)

Data Lines = full width, word split

Overflow Lines = indented three characters

Page Overflow = full record

E. KEY IDENTIFIER (CONTROL FIELD):

Not Applicable

F. RECORD LENGTH:

Not Applicable

G. BLOCKING FACTOR:

Not Applicable

H. PURPOSE:

To display the contents of the data lines comprising a stored strategy.

TOEIC H.4 - IMMEDIATE COMMANDS

A. DATA SET NAME:

Strategy Names Display Format

B. CREATED BY:

NDBSTRT

C. TYPE OF FILE:

Screen Display

D. ORGANIZATION:

Data Lines = complete 16 character strategy names
separated by two blanks (as many as will fit on a
line).

E. KEY IDENTIFIER (CONTROL FIELD):

Not Applicable

F. RECORD LENGTH:

Not Applicable

G. BLOCKING FACTOR:

Not Applicable

H. PURPOSE:

To display the names of the strategies present in the
strategy data set.

TOEIC H.5 - INTERNAL PROFILE TABLE

A. DATA SET NAME:

NASIS.PROFILES

B. CREATED BY:

NTSPRO

C. TYPE OF FILE:

Sequential Array

D. ORGANIZATION:

Sequent-1	(Synonyms)	-	Sequential
Sequent-2	(Defaults)	-	Sequential
Sequent-3	(Default Data)	-	Random

E. KEY IDENTIFIER (CONTROL FIELD):

Not Applicable

F. RECORD LENGTH:

1500 Bytes

G. BLOCKING FACTOR:

Not Applicable

H. PURPOSE:

To save user defined synonyms and defaults.

TOEIC H.6 - IMMEDIATE COMMANDS

A. DATA SET NAME:

NASIS User Profile Dataset NASIS Profiles

B. CREATED BY:

Not Applicable

C. TYPE OF FILE:

Members are SAM, PDS containing all BPAM members.

D. ORGANIZATION:

Sequential

E. KEY IDENTIFIER (CONTROL FIELD):

Not Applicable

F. RECORD LENGTH:

F(2000)

G. BLOCKING FACTOR:

Not Applicable

H. PURPOSE:

To contain the lists of user defined synonyms and defaults for a particular NASISID.

TOEIC H.7 - COMMANDS FOR DATA RETRIEVAL

A. DATA SET NAME:

VERBTAB Table.

B. CREATED BY:

The NASIS modules which prompt for commands.

C. TYPE OF FILE:

Table

D. ORGANIZATION:

PL/I Data Structure

E. KEY IDENTIFIER (CONTROL FIELD):

Not Applicable

F. RECORD LENGTH:

Not Applicable

G. BLOCKING FACTOR:

Not Applicable

H. PURPOSE:

1. VERBTAB EXTERNAL CONTROLLED.

This table contains the information necessary to associate a set of valid verbs (commands) and their respective entry points.

2. #_ENTRIES BINARY FIXED.

This field contains the count of the number of valid entries in the list below.

2. SIZE BINARY FIXED.

This field contains the count of the number of entries that can be contained in the list below.

2. SYMBOLIC_ID CHARACTER (8).

This field contains the default symbol that can be used to define user written extensions to this list of verbs.

2. COMMAND (VERB_COUNT).

This list is used to describe the commands recognized by the defining module.

3. NAME CHARACTER (8).

This field contains the command name.

3. ROUTINE CHARACTER (8).

This field contains the name of the entry point to be called when this command is entered.

1. VERB_COUNT BINARY FIXED.

This field must be set to the maximum number of entries allowable in the verb list, before the table is allocated.

I. PL/I DECLARATION:

```

/*          GENERALIZED NASIS SYSTEM VERB TABLE          */
DCI 1 VERBTAB EXT CONTROLLED, /*DEFINE THE VERB TABLE    */
  2 *_ENTRIES BIN FIXED, /*DEFINE THE CURRENT SIZE     */
  2 SIZE BIN FIXED,      /*DEFINE THE MAXIMUM SIZE     */
  2 SYMBOLIC_ID CHAR(8), /*DEFINE THE DEFAULT TERM     */
  2 COMMAND(VERB_COUNT), /*DEFINE THE VERB ENTRIES     */
    3 NAME CHAR(8),      /*DEFINE THE VERB NAME        */
    3 ROUTINE CHAR(8);   /*DEFINE THE ROUTINE NAME     */

DCI VERB_COUNT BIN FIXED; /*DEFINE THE TABLE SIZE     */

```